$4.95

HYPER LINK MAGAZINE

June/July 1988

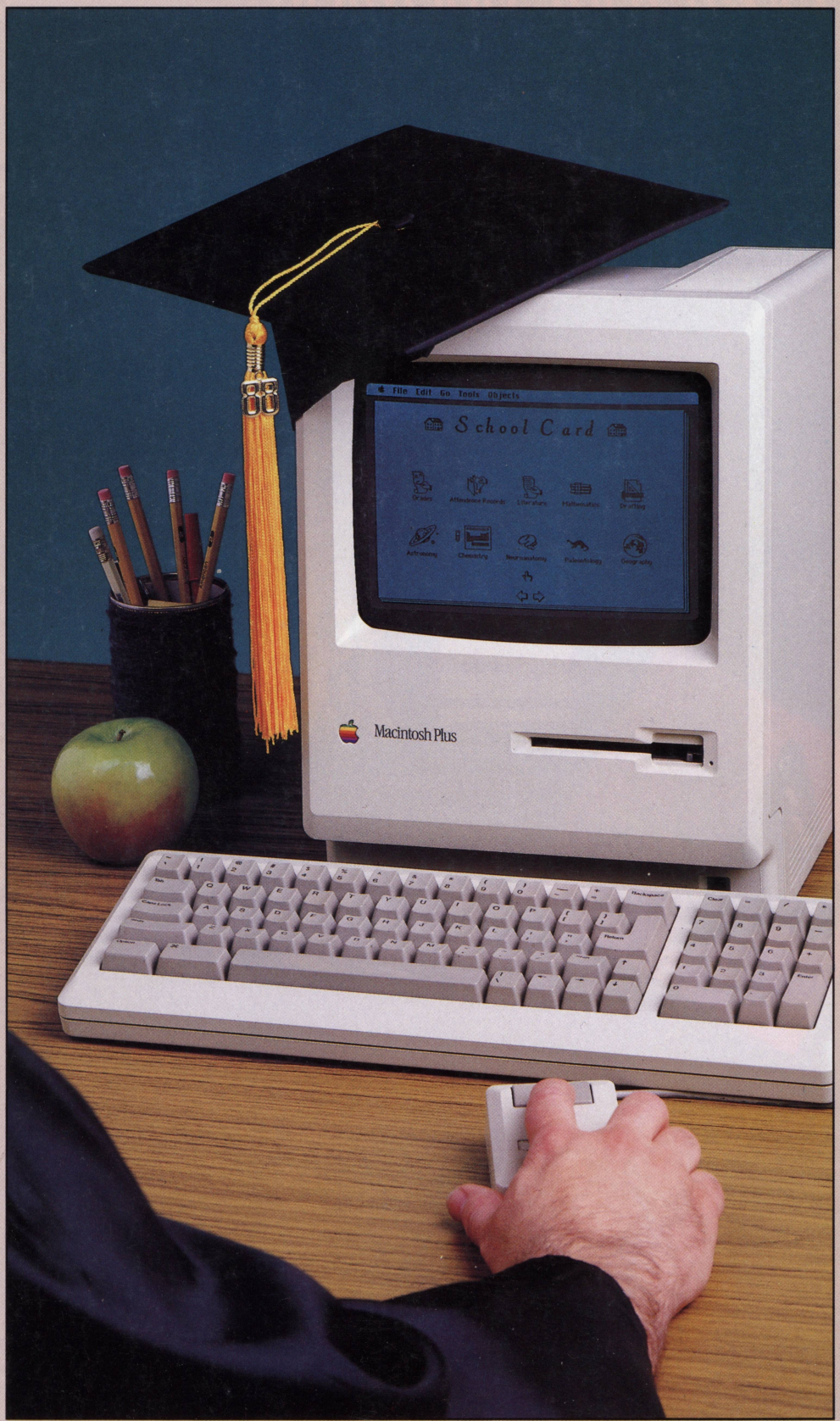
Danny Goodman on HyperCard 1.2

Educators Track Classes with HyperCard Stacks

HyperCard Stack Teaches About the Brain

Enhancing HyperCard Printing Facilities

And Much More!



06
0 74470 74465 8

MacRecorder.™

Open Your Macintosh™ to the World of Sound.

THINK of the power you would add to sales presentations, slide shows, and reports if you could add narration, music, or special sound effects to them.

MacRecorder turns your Macintosh into an audio workshop.
The MacRecorder Sound System includes *MacRecorder* sound digitizer hardware, *SoundEdit*™ sound engineering software, and *HyperSound*™ HyperCard™ stackware.

Record real live or prerecorded sound into your Macintosh in mono or stereo through the MacRecorder. Edit sound and create special sound effects with SoundEdit. Save your recordings in HyperCard stacks, instrument files, or system resources. MacRecorder is compatible with VideoWorks™, Studio Session™, Jam Session™, and SoundCap™ program formats. You can also compress sound in real time to save RAM and disk space.

It's as easy as it sounds. And it sounds as good as FM radio.

Let your presentation speak for itself.
Mix voice and music to create soundtracks for business presentations, demo disks, and training tapes. With immediate applications in marketing, advertising, sales, and personnel training, MacRecorder is a valuable tool for your business.

Call your local dealer to hear more about the MacRecorder Sound System.

farallon computing

2150 Kittredge Street, Berkeley, California 94704
415 849.2331 Fax 415 841.5770

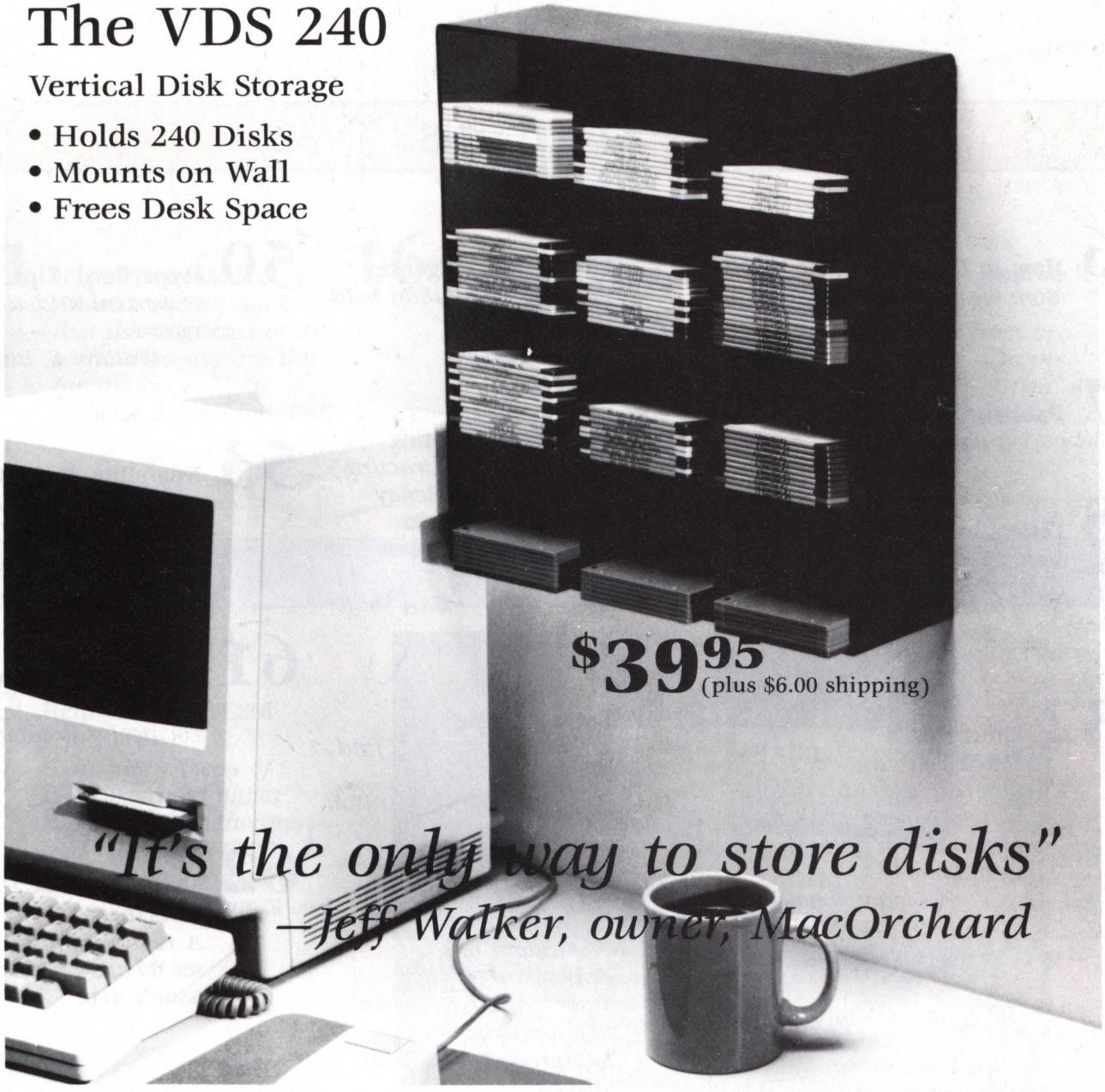
Circle 150 on reader service card

Reclaims Desk from Disks

The VDS 240

Vertical Disk Storage

- Holds 240 Disks
- Mounts on Wall
- Frees Desk Space



$39.95 (plus $6.00 shipping)

"It's the only way to store disks"
—Jeff Walker, owner, MacOrchard

To Order: 1-800-942-4008

Dust covers and office partition hangers available.
7 days COD/Visa/MC/Amex • Government, Education PO's accepted.
Call for volume pricing.

Vertical Solutions • P.O. Box 7535 • Olympia, WA 98507 • 206.352.2097

— We Stood Disk Storage on End.

HYPERLINK MAGAZINE

COLUMNS

FEATURES

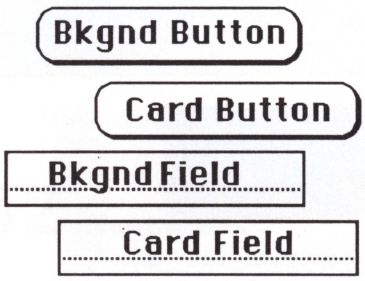
REVIEWS

Putting HyperLink buttons, fields, and stacks into your Mac

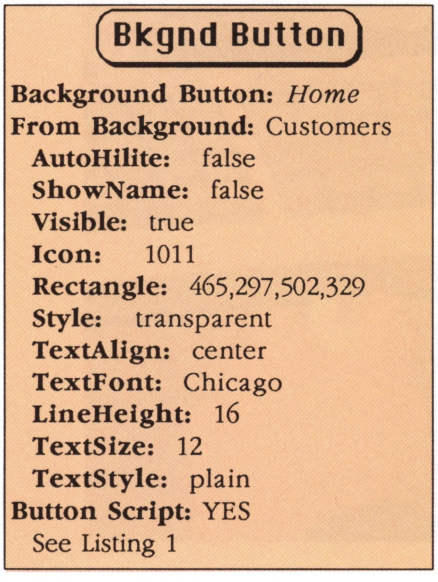
How to Use HyperLink

The articles in *HyperLink* are designed to be the best compromise between the use of page space and the graphic representation of what you see on your Mac's screen. In addition, we have tried to make it very easy for you to enter *HyperCard* objects (buttons, fields, etc.) without errors.

For instance, in a button or field description the icon label clearly shows whether the object is in a background or a card layer:



HyperCard objects have a description and possible *HyperTalk* script associated with them. Here's the standard description format:



The description contains all the information to create the object in *HyperCard* on your Macintosh. You will note that the last attribute in the foregoing button description indicates the existence of a script. The object's associated script is listed in a window just like the one on your Mac. The "elevator shaft" and the "white elevator car" indicate the position in the script that is showing in the window:

Listing 1 - Script for background button *Home*

```
put 0 into result
repeat until counter = 0
  get last char of mask
  put it into temp1
  delete last char of ¬
  mask
  get last char of CC
  put it into temp2
  delete last char of CC
```

As you can see, the scripts are visually easy to check as they are entered because they look the same as what appears on your Mac's screen.

Notice the hook character (¬) at the end of line five in the listing (obtained within the script editor window by holding down the Option key and pressing Return). It indicates that the current line continues on the next line. If you enter the line as shown it will work properly. If you find there is enough room for the entire statement to fit on one line when using the HyperCard script editor, this character may be omitted. In fact, these characters should only appear at the end of a physical line, otherwise a syntax error will result.

Making it Easier

HyperLink Magazine's staff has created a special *HyperCard* stack called *StackFramer*™, (available on the Volume 1, No. 1 *StackSolutions* disk). This stack makes it easy to enter stacks from our pages. *StackFramer* asks you questions about the object you are entering and leads you through the process of building screen displays identical to those in the magazine.

By visually verifying a button's or field's description character-by-character, line-by-line on the screen with the description on the page, you know it is error free.

StackFramer incorporates a special script entry window for you to enter scripts. This script entry window allows full use of the Edit menu tools unlike the actual *HyperTalk* script editing window.

After you have given *StackFramer* all of the object definitions (stack, backgrounds, cards, buttons, and fields) and object scripts, it generates the stack for you. You may generate as many copies of the stack as you desire.

If you save the *StackFramer* copy containing the template of a stack, you can later make changes to the template's object descriptions or scripts and generate new stacks that include these changes.

The Easiest Way

If you just don't have the time or inclination to hand enter all of the *HyperCard* objects and stacks, consider purchase of the *HyperLink StackSolutions*™ disk for that magazine issue. Everything is there including special sounds and all graphics.

HYPERLINK MAGAZINE

Jun/Jul 1988 • Volume 1 No. 2

Publisher
David G. Brader

Editor
Roger Wood

Contributing Editors
Joan Donaldson
Victor S. Johnston, PhD.
Carol Kaehler
James Paul
Dan Shafer

Copy Editing
Ruth Gibian

Director of Design
Marv Boggs

Manager of R & D
William K. Balthrop

Controller
Mark Andersen

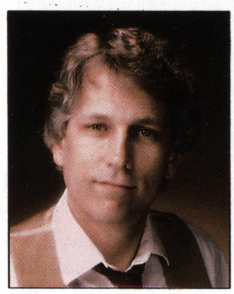
Public Relations
Ken Jackson

Advertising Sales
Carole Eversole
(503) 484-5157

HyperLink Magazine is published 6 times a year by Publishers Guild, Inc., P.O. Box 7723, Eugene, OR 97401. Telephone (503) 484-5157.

Editorial material should be submitted to HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401. Submissions will not be returned.

Publisher's Card



Well, a lot of events have occurred since our premier issue hit the marketplace two months ago—Apple has released a new version of *HyperCard* (version 1.2) with some improvements and many bugs exterminated, Danny Goodman has a new book out, *HyperExpo* came into being, and, of course, this second issue of *HyperLink* was created.

Taking the last topic first, you will note that two of the features in this issue are directed to education. If you are involved in the field of education *Class Stacks* and *Hyper-NeuroAnatomy* will be of special interest. Even if you are not an educator, spending time with these articles will enlighten you and activate your creativity. As an example of what I mean, *Class Stacks* could be used with minor modifications by a butterfly collector to track a collection. The classroom seating chart becomes the butterfly specimen display case, The student record card becomes the specimen record card, etc.—use your imagination.

While we are on the subject of *HyperLink* itself, the staff and I want to thank all of you for your support and comments. When we put our collective hearts and souls into a publication like *HyperLink Magazine* it is very rewarding to see and hear the overwhelmingly positive responses from you, our readers.

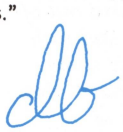
Yes, we listen to your negative comments, too. You told us the subscription order form was confusing. You told us about some small points in the printed stack listings that should be improved. These are now fixed. You told us you wanted combination *HyperLink Magazine/StackSolutions Disk* subscriptions. These are now available (see the order form inside the back cover).

We need your communication to keep *HyperLink* on track with your needs. There are two methods that are available for this communication. The first, and most obvious, is the *Letters to the Editor* column in the magazine. Keep those cards and letters coming! The second is to come visit us at one of the shows in your area. If you are in San Francisco June 11th or 12th, come to the Civic Auditorium and see us at the new *HyperExpo & StackMart*. See the advertisement on page 33 for more information on this show. If you are in the Boston area during the *MacWorld Expo*, August 11th through the 13th, come see us at booth 157 in the Bayside Expo Center.

Apple has released *HyperCard* version 1.2 which has improvements that are not immediately apparent to casual users. The changes are significant to those of you that build your own stacks, however. Some of these changes are discussed in Danny Goodman's contribution this issue. This feature covers the new string-search changes and consists of two chapters excerpted from Danny's new book *Danny Goodman's HyperCard Developer's Guide* published by Bantam Computer Books.

One of the *HyperCard* areas that needs improvement is the facility for printing information that resides in a stack or stacks. Two commercially available solutions are reviewed in this issue. The pros and cons are discussed by Roger Wood, *HyperLink's* Editor. Do take the time to explore *HyperCONTROL* and *Reports* with Roger.

Next issue we will delve into building your own hypertext tool, doing interactive (graphic) fiction tools for personal productivity, and having fun in *HyperCard*. Until then, be creative and enjoy the world of *HyperLink Magazine* "The One Essential Resource for HyperCard Users."



David G. Brader

Letters to the Editor

Some Suggestions

Bravo! I have anxiously awaited the availability of a serious magazine to cover the *HyperCard* realm.

In your premier address to the *HyperCard* community, you welcomed criticisms of *HyperCard* (and by association, third party developers)—so, here are two:

1. For Apple: *HyperCard* supports buttons and fields. It would be wonderful if it also supported graphic windows; that is, entities that contain graphics which could be hidden and shown, positioned by coordinate, loaded from a file, etc. Currently, I find that I need to play too many card switching games, causing much clutter on the background.

2. For third party developers: How about a record manager (such as Btrieve in the DOS world) that was accessed by an XCMD? Or an XCMD interface for the Mac database products? If there is one out there, the author/distributor isn't promoting loudly enough.

Thanks for the opportunity to speak. I look forward to a lengthy relationship.

Stuart Malin
Chicago, IL 60611

Thanks for the comments and criticisms, Stuart—Apple does seem to be listening. Although the latest version of HyperCard (1.2, which should be available by the time you read this) does not implement windows for pictures, it does incorporate commands that allow for better management of pictures within stacks—e.g., hiding and showing pictures.

The Import and Export buttons, introduced with HyperCard 1.1, help with the moving of data in and out of HyperCard, but are still rather limited. There are some stacks on bulletin boards that import specific types of data—Import SYLK 1.0, for example. A stack that knows several formats might be a great project for someone out there. . . hint, hint.

Listing - Margaret Child's version of last issue's "Field of Intelligence."

```
on mouseUp
  global clickLine
  put "Customer List" into fName
  put textHeight of field fName into fHeight
  put round(scroll of field fName/fHeight) into offTop
  --it makes more sense to calculate the number of lines
  --"off the top" hence the name offTop
  --scroll is approx multiple of 'fheight'
  put item 2 of the clickLoc - item 2 of rect of field fName ¬
  into Vir
  put round(Vir/fHeight+.5) into lineLoc
  put offTop + lineLoc into clickLine
  put "clickLine" && clickLine
end mouseUp
```

Not-So-Intelligent Field?

I'm sure you know by now that the example of an intelligent field in the April/May 1988 issue of *HyperLink* (p. 30) does not work. There are two problems—one syntactical and one logical.

The syntactical error is that your example showed a superfluous parenthesis before "fName" in the statement:

```
put round(scroll of
field fName ...
```

The logical error has to do with overkill—adding .5 and rounding when calculating `topLine`.

In order to keep the logic straightforward, it makes most sense to me to think in terms of how many lines have been scrolled off the top of the field and then to add this value to the current line number. Checking the calculation for the number of lines "off the top" is somewhat tricky since the value of scroll varies depending on how the field is set up and whether you scrolled using the scroll arrows or the sliding box. (With my second line showing as the first line of the field, for a text height of 16, one scroll value was 16 after clicking once on the down arrow. When I repositioned the same line in the same place by clicking in the scroll bar, the value was 13). The "off the top" calculation can simply be:

```
put round(scroll of ¬
field fName/fHeight) ¬
into offTop
```

The mathematics to calculate "the line in which the click occurred" requires trapping a value between an upper and a lower limit (i.e., if the text height is 16 and the click occurs between 0 and 16, then it occurred on line 1; between 17 and 31 it occurred on line 2, etc.). Your method works well for this.

Please find enclosed a copy of my version of the intelligent field. I hope I never have to spend as many hours on so few lines of code again!

Margaret Child
San Francisco, CA

Right on both counts, Margaret. The first error was a typo that was not caught in final editing. We're sorry for the inconvenience this has caused, and we'll try to keep such HyperTypos from interfering with our coverage of HyperTalk scripting. The logical error you mention is not so much an error as overkill, as you point out.

We have discovered another problem associated with using the scroll property to find the location of text in a field. The method employed by this script only works if each line in a field has a Return character at the end. That is, HyperCard only recognizes a string of characters as a line if it is terminated by a Return. When a line is longer than the width of a field, the text wraps to the next line, but the scroll property only gives you the number of Return characters "off the top." As long as there are Returns at the end of each line, the routine works fine. But for every line off the top that wraps without a return, the number in clickLine is reduced by one. Any readers with a workaround for this bug?

Credit Where It's Due

Thank you for including *Dan Shafer's Script Expert* in your "Survey of HyperCard Resources" in your premier issue. Both Dan Shafer and I were credited with writing the program and I'd like to

give credit where credit is due. Dan is the author and creator of *Script Expert*. He wrote every line of code in the program, and credit for the concept should go to him as well. Another unsung hero in the project is Tomas Hernandez, a brilliant artist who created the crisp graphics that make *Script Expert* so visually exciting. My role as publisher has been more of a coach and provocateur than program author—though as all publishers, I couldn't sleep at night if I didn't suggest some features to Dan. Dan, with Tomas's help, deserves full credit for producing a revolutionary program.

One other minor correction: *Dan Shafer's Script Expert* has always been priced at $79.95 suggested retail. You mistakenly listed it at $69.95. Hyperpress also markets *Icon Factory* (by James Paul) for $49.95, *HyperSpell* (with Microlytics/XEROX technology) for $79.95, and the *HyperTalk Pocket Reference* for $11.95.

David Gewirtz
President/Publisher
Hyperpress Publishing
P. O. Box 8243
Foster City, CA 94404

Thanks for the info, David. Sorry for the mistake on the pricing, I hope you haven't been inconvenienced. Also, thank you for keeping us up to date on your company's latest releases, we will defintely look carefully at these new stracks

Survival Request

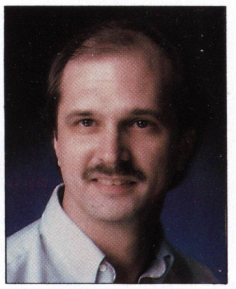
Please survive! I need you! Oh, and here's a tip: Danny Goodman's book (which I am surely not alone in plugging through) fits perfectly in a transparent cookbook holder.

A. K. Westmoreland
Seattle, WA

I hope this issue provides assurance of our intention to survive, A.K. Sounds like you've found a great recipe for getting the most from your HyperCooking.

Order Entry Database Praise

I received my first copy of your great new magazine on Saturday. I must tell you that I am very pleased with it, and

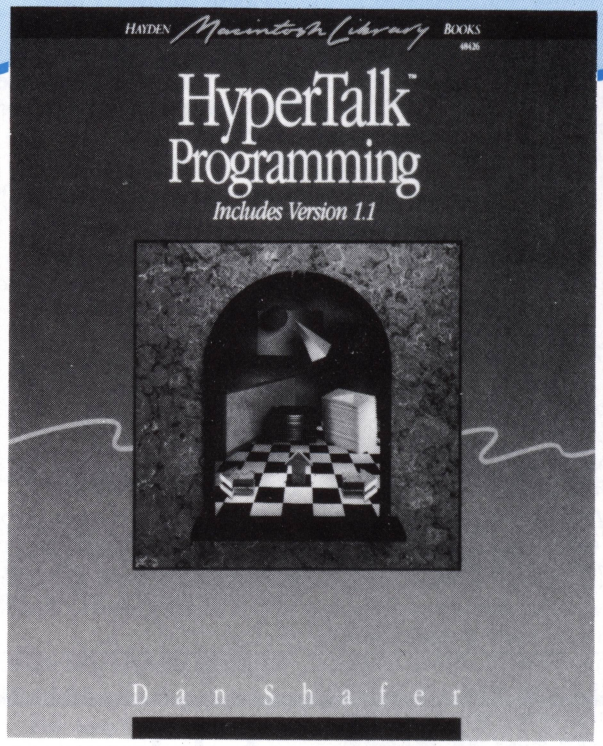


the disk that was enclosed. I have read the magazine from cover to cover twice and have already found a program that will help me with my small business. It is the "Order Entry Database." It is just what I have been looking for to keep track of my various customers. This program alone will be worth the price of the whole year's subscription.

All the articles and programs in this issue are of interest, and I shall probably have use for several of them

—continued on page 62

A Programming Guide for All HyperCard™ Users



Covers Every Aspect of HyperTalk™ Programming

- Discusses the background and concepts behind HyperTalk and how they fit into the programming environment
- Provides powerful programming aids that assist in the design and development of HyperCard applications
- Includes the creation and incorporation of language extensions using XCMD and XFCN with instructions on how to add commands in C or Pascal
- Shows how to add sound, graphics, and communications to HyperTalk scripts

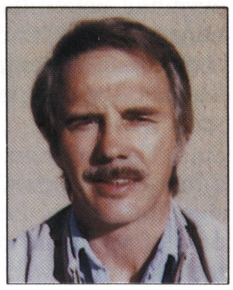
This comprehensive tutorial covers every feature of HyperTalk, the language of HyperCard. Coverage includes everything from the concepts behind HyperTalk, object-oriented programming, to sample scripts for ready-made applications. Details on how to extend HyperTalk with C and Pascal programming are fully presented, along with use of graphics, sound, communications, and much more. Two complete scripts for applications are provided including one which enables "semi-automatic" programming. Three appendices provide reference information such as the complete listing of HyperTalk commands, functions, operators, key words, identifiers and much more. A special pull-out card provides a quick reference to HyperTalk commands and functions. *Hypertalk Programming* reveals the power of HyperCard and shows you how to utilize it. Get your copy today!

Visit your local book retailer or call

800-544-0339

or charge your VISA, MasterCard, or send a check for $27.00 postage paid (per book) to HyperLink Magazine, P. O. Box 7723, Eugene, OR 97401

Exploring the Hyperlinked Brain



Hyper-Neuroanatomy

by Victor S. Johnston, Ph.D.

The "Hyper-Neuroanatomy" stack is a commercial product written and copyrighted by Dr. Victor S. Johnston. It is available for $25 from Kinko's Courseware or from *HyperLink Magazine* at (800) 544-0339. A special "Hyper-Neuroanatomy Demo" version of the stack, featuring the "Medial Temporal" section and all cards related to this section from the System and Information Levels, is on this issue's *StackSolutions* microdisk.

It is only when we are confronted with such unusual cases as Dr. P, *The Man Who Mistook His Wife for a Hat* (by Oliver Sacks, Harper and Row, New York, 1987), that we are reminded of the extent to which our view of reality is dependent upon the physical structure and chemical integrity of the human brain. Replace one chemical with another and we may see sound or hear light; a small lesion may leave us unable to learn a single new fact for the rest of our lives. The human brain, the seat of our thoughts, feelings, passions and desires, stands alone as the most formidable challenge facing scientific investigation. Weighing a mere 3.5 lbs, but rich in its intricate complexity, it beckons young neuroscientists like a siren, and captivates them for life by its ever enticing mysteries.

Almost 25 years have passed since my neuroanatomy professor opened a large plastic container, marked MORGUE, and placed its contents on the desk in front of him. I remember the smell of formaldehyde and the mixed feelings of awe and queasiness as he peeled away the dura matter and cut through the soft tissue with a knife to expose the inner sanctum of the Cerebrum. It was a difficult course. We saw many slides and models, and made innumerable drawings as we learned the new vocabulary which would allow us to discuss the inner secrets of the "enchanted loom." Now, 25 years later, I am that professor, faced with the same formidable task of teaching the structure and function of the human brain.

Weighing a mere 3.5 lbs, but rich in its intricate complexity, the human brain beckons young neuro-scientists like a siren, and captivates them for life by its ever enticing mysteries.

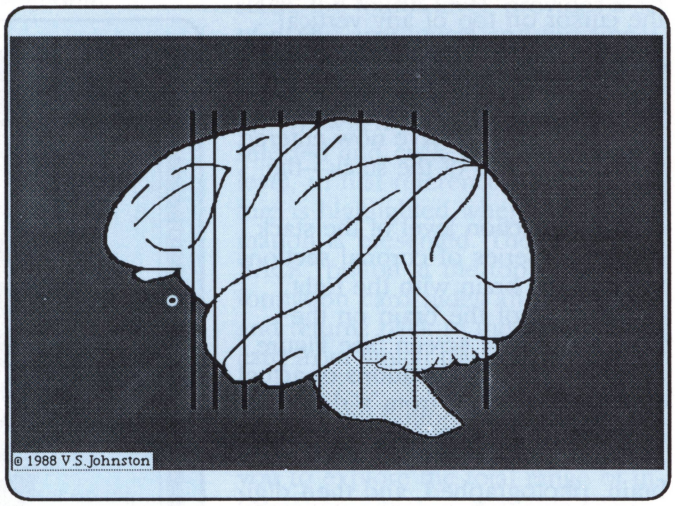


Figure 1

The opening card of the "Hyper-Neuroanatomy" stack. Clicking on one of the lines takes the user to a digitization of a cross-section of the brain.

Why Use *HyperCard?*

I was immediately attracted to *HyperCard.* Perhaps it was because, like the human brain, *HyperCard* allows knowledge to be stored and retrieved by association, rather than by hierarchies or indices. I bought a copy of Danny Goodman's *The Complete HyperCard Handbook* and set about the "fearful task" of learning yet another computer language. But work soon turned into fun. The brain sections stored in my Mac became

Dr. Victor S. Johnston is Professor of Psychology at New Mexico State University. He received his doctorate from the University of Edinburgh, and did post-doctoral work at Yale and Stanford.

alive as I linked them to other sections and data. Encouraged by the enthusiasm and support of my colleagues, the "Hyper-Neuroanatomy" stack was born.

The program has evolved into a four-level *HyperCard* stack designed to teach the basic neuroanatomy of a primate brain. It allows the user to browse through the brain, organize structures into systems, and request anatomical or functional information about any specific structure or system.

The Whole Brain view is the top level of the stack (see Figure 1). This shows the lateral aspect of a primate brain with the front of the brain (anterior) to the left, and the posterior to the right. A series of eight vertical lines indicates strategic locations where the brain may be sectioned in order to view its internal structure. To cut the brain at any desired location, you simply place the cursor on top of any vertical line and click. You will then see a transverse (coronal) section through the brain at the location you selected. You have now entered the second level of the stack—the Section level.

The Section level of the stack displays a series of coronal sections through the brain with the right hemisphere of the brain on the right side of the screen (see Figure 2). These were digitizations of actual sections of a brain from a rhesus monkey. The sections were dyed to clarify the different parts of the brain, photographed, and then digitized using *ThunderScan.* The *ThunderScan* files were saved in *MacPaint* format, and the final cleanup was done in *Full Paint.* [For a detailed description of using *ThunderScan* to import graphics into *HyperCard* see "Scanning the Arts" in *HyperLink Magazine* Vol. 1, No. 1—Ed.]

The dark line in the small brain on the upper-right of the screen shows the position of the section you are currently viewing. Clicking this smaller picture will take you back to the Whole Brain level of the stack.

There are a large number of transparent buttons on the right side of each coronal section (see Figure 3) that can be located (as in any *HyperCard* stack) by holding down the Command and Option keys at the same time. I kept the buttons transparent so the user would get used to looking at a clear image of the brain instead of one cluttered with buttons. Clicking any of these transparent buttons advances you to the System level of the stack. The arrow buttons, in the upper right and left of this screen, provide a means for moving through the coronal sections one at a time. These allow you to trace a known structure through several sections of the brain. [This is the only section included with the "Hyper-Neuroanatomy Demo," and these buttons will actual take you to the fi-

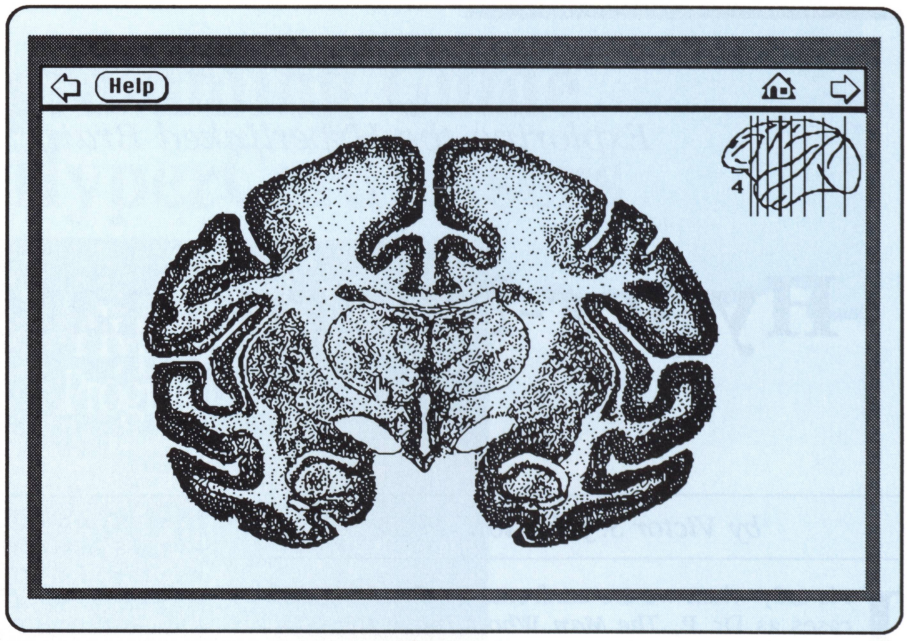


Figure 2

The "Medial Temporal" card of the stack. This was created by using ThunderScan to digitize actual photographs of dyed cross-sections of the brain of a rhesus monkey.

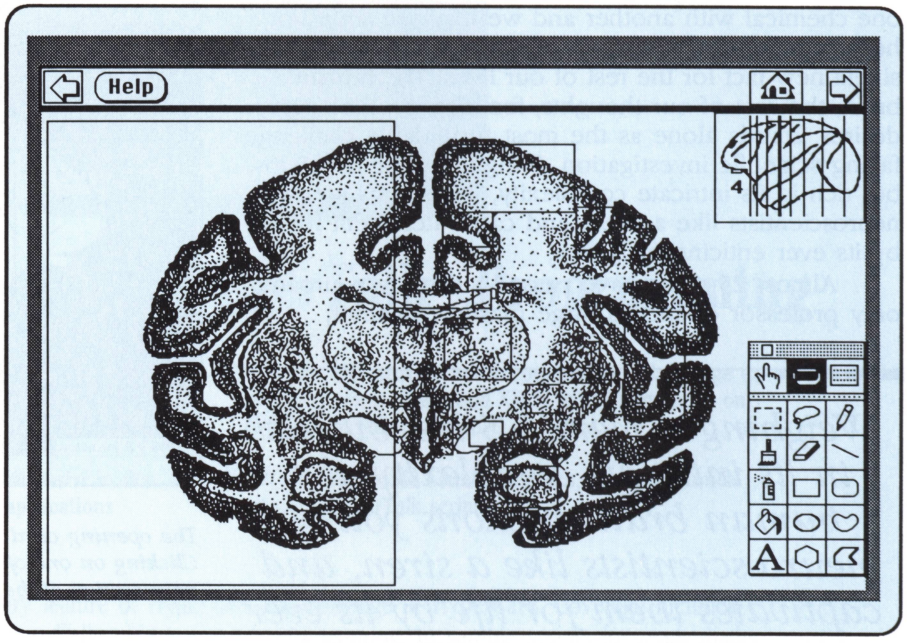


Figure 3

This is the same card as shown in Figure 2, but with the button tool selected to show the location of the transparent buttons. All navigation buttons are located on the right side of the drawing. The large button on the left simply displays a message informing the user to click on the right-hand side.

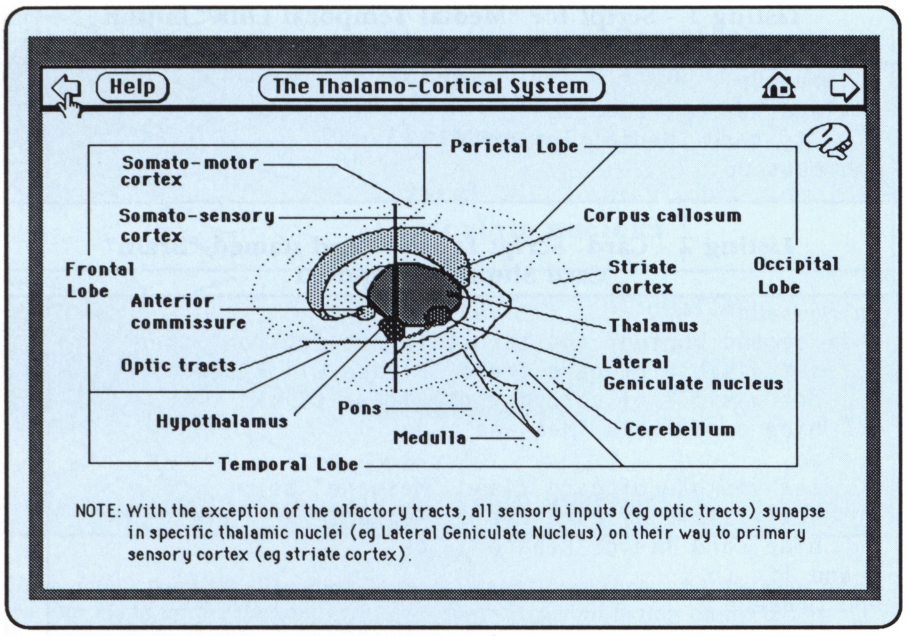


Figure 4

This is the "Thalamo-Cortical System" card—one of five cards that makeup the System Level of the stack. Clicking on any of the bold words activates a button that takes you to the Information Level card that discusses that part of the brain.

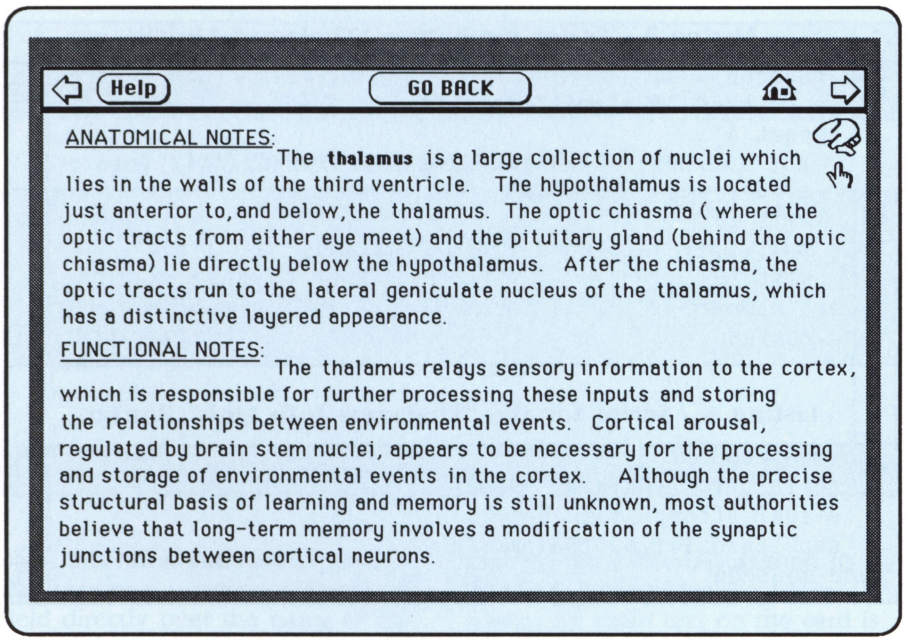


Figure 5

A card from the Information level of the "Hyper-Neuroanatomy" stack. The text on the card was produced using the text tool under the Tools menu. The bold text, "thalamus," at the top center of the card, is in a text field that is only made visible when the user clicks on the word "thalamus" in the System level of the stack, shown in Figure 4.

nal (Information) level in the demo version of the stack—Ed.]

The System level of the stack groups the major brain structures into a small number of anatomical or functional systems. (One such system, the Thalamo-Cortical System, is shown in Figure 4.) When the user enters the System level by clicking on a structure in a coronal section, the screen displays the system to which the structure belongs and flashes the name of the selected structure on the card. A vertical line on this card is used to indicate where you have cut through the system. Clicking on this line will return you to the coronal section.

Five major anatomical and functional systems are identified at the System level. These are:

- The Major Sulci
- The Basal Ganglia
- The Limbic System
- The Ventricular System
- The Thalamo-Cortical System

The major system components are labelled in card buttons at the top of each card. When you click one of these buttons, you are taken to the Information level of the stack. In addition, the arrows in the upper corners of the system cards allow you to move through any system in an anterior or posterior direction.

The Information level of the stack, the fourth level, provides a set of useful anatomical and functional notes about systems or structures selected at the System level (see Figure 5). When a specific structure is selected and clicked at the System level, th first reference to that structure is highlighted when the information is presented. The "GO BACK" button at the top of an Information card hides all the fields and returns back to the System level card from which this card was entered.

Arrow buttons in the upper corners of information cards allow you to explore the total range of information which is available. You will find that a little functional information about a structure will help in remembering its anatomical location and relationship to other structures.

The Scripts

I did not want to clutter the simple display of the brain in the opening card of the stack (see Figure 1). I felt that the user would naturally click somewhere on the brain in the picture. If the click occurs on one of the vertical lines, one of the transparent buttons covering the lines invokes an `on MouseUp` handler like the one in Listing 1, and the user is taken to a cross-section like the one in Figure 2. When the user clicks the mouse anywhere else

on the screen, instructions appear telling the user to click on one of the lines. I display the message to "Please click on any vertical line to section the brain" in a card field called "Please Click."

In addition, this special demo version requires another hidden field called "Message," which explains how to obtain the full functioning version of the "Hyper-Neuroanatomy" stack. Because only the fourth section is implemented in the demo version of the stack, I wanted to display this message if the user clicks on any of the vertical lines except the fourth one.

The main handler to display these messages is in Listing 2. The key to this routine is using `the target`—a *HyperCard* variable that contains the name of the object on which the user clicks. If `the target` turns out to be the button covering the fourth line from the left, the `on mouseUp` handler in Listing 1 takes care of the action, and the user is taken to the "Temporal Medial" card shown in Figure 2. Otherwise, the `mouseUp` message proceeds up the *HyperCard* hierarchy to the Card script that contains the handler in Listing 2.

This handler first checks to see whether the user has clicked on something other than one of the buttons, in which case the "Please Click" field is activated. The script makes use of a device programmers often call a "toggle"—so named because it operates like a physical toggle switch. It works like this: If the field is visible, hide it, but if it is hidden, make it visible. I use a property of fields called `visible` to accomplish this. The `visible` property is a boolean variable (i.e., either true or false), and the handler simply sets it to the opposite value. Thus, it shows the field if it is hidden, or hides it if it is visible.

The `else` part of this handler is invoked if the `the target` is a button. Here the toggle is used again, this time hiding or showing the "Message" field. Notice that I also hide the "Please Click" field in this case, because I know that if we get to this part of the handler the user did click on one of the buttons. I also include the `on CloseCard` handler at the bottom of the script to ensure that both fields will be hidden the next time the user returns to this card.

Listing 1 - Script for "Medial Temporal Link" button on Card the "Brain"

```
on mouseUp
  visual effect zoom open
  go to card "Medial Temporal"
end mouseUp
```

Listing 2 - Card Script for the Card named "Brain" Card Shown in Figure 1

```
on mouseUp
  if second word of the target is not "button" then
    set visible of card field "Please Click" to ¬
    not visible of card field "Please Click"
    hide card field "Message"
  else
    set visible of card field "Message" to ¬
    not visible of card field "Message"
    hide card field "Please Click"
  end if
end mouseUp

on closeCard
  hide card field "Please Click"
  hide card field "Message"
end closeCard
```

Listing 3 - Script for button "Thalamus Link"

```
on mouseUp
  go to card "Thalamo-Cortical"
  repeat 4
    put XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX into ¬
    card field "Thalamus"
    wait 10
    put empty into card field "Thalamus"
    wait 10
  end repeat
end mouseUp
```

Listing 4 - Script for the "Thalamus Info Link" Button

```
on mouseUp
  go to card "Thalamus Info"
  visual effect zoom open
  show card field "thalamus"
end mouseUp
```

Highlighting Text

In the process of writing the "Hyper-Neuroanatomy" stack, I developed some scripts which I found to be useful for drawing attention to a small segment of information when it is embedded in a larger body of text.

These scripts for temporary (flashing field) or more permanent highlighting (highlighted fields) are shown in Listings 3 and 4 respectively.

The body of text on the System level cards was entered with the text tool, so it is not text in a field, but text as bit-mapped graphics. I placed transparent fields directly over the words to be highlighted. When the user clicks on a one of the transparent buttons on a Section, like the one in Figure 3, an `on mouseUp` handler like the one in Listing 3 is invoked. This transports the user to the proper System level

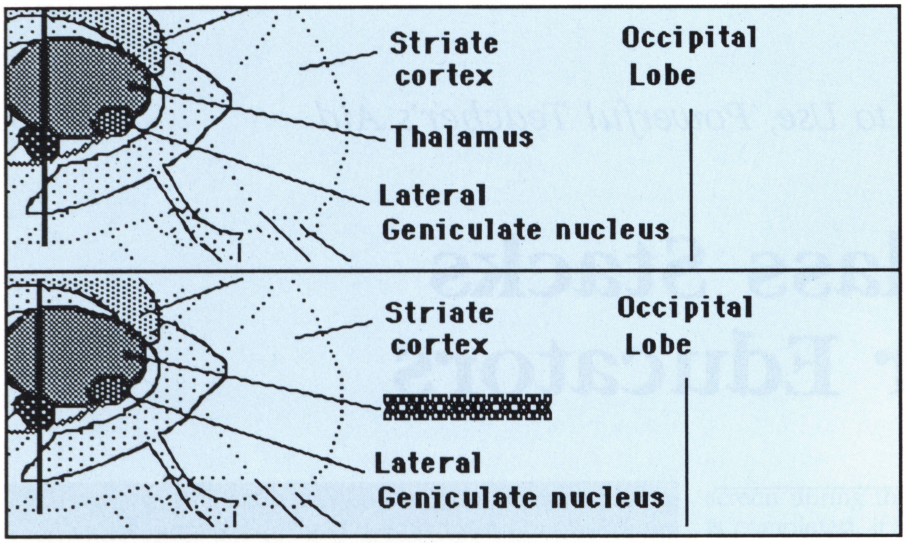


Figure 6

These details from the "Thalamo-Cortical System" card show what happens when the user clicks on the thalamus (actually the transparent button on top of the thalamus) in the Section Level. First, the script in Listing 3 takes the user to this card. The text, "Thalamus," is alternately shown (upper drawing) and covered with Xs (lower drawing). The alternation is done rapidly enough to make it easy to pick out the name and the location of the Thalamus on the System Level card.

Listing 5 - Script for Button "GO BACK"

```
on mouseUp
  goBack
end mouseUp
```

Listing 6 - Script for Background "Main"

```
on goBack
  set cursor to 4
  set lockscreen to true
  repeat with fieldNum = 1 to number of card fields
    hide card field fieldNum
  end repeat
  go back
  set lockscreen to false
end goBack
```

card, and then alternately places and removes a group of Xs into this field directly over the name of the part of the brain that was clicked on back at the Section level (see Figure 6). Notice that the Xs are in an outline font to make them stand out. By doing four repeats of this placing and removing process, as shown in Listing 3, the user is made instantly aware of both the name and the location of the part of the brain that was selected at the Section level.

I used a different method of highlighting text on the Information level cards. Because all text within a field has to be in the same font, I devised the following method to highlight a specific piece of text. Again, the main text on the card is bit-mapped and not in a field, having been entered using the text tool. I created an opaque card field on top of each part of the text that I wanted to be able to highlight. Then I entered text into this field that was identical to the text underneath it, but I used a bold font in this field so it would stand out next to the non-bold text already on the screen. I positioned it so that it exactly overlays the information to be highlighted. You can see the effect attained by looking at the word "thalamus" in the upper center of Figure 4.

It is important that only the one word is highlighted when this card is opened so the user has immediate feedback on the connection between the text and the section of the brain that was clicked. To ensure this, all the fields on the card are hidden when the user clicks the "GO BACK" button. I was able to consolidate this into a single custom `goBack` handler placed in the background script. Notice that *HyperCard's* regular `go back` handler has a space in the middle, so I can still make use of this `go back` routine if I name mine without the space (see Listing 5 and 6). This handler in listing 6 will `hide` all the card fields and then call the regular *HyperCard* `go back` handler.

An important point to note: Much of the linking of buttons to cards within this stack was not done by direct scripting. All I had to do was create the button using the "New Button" command, open the Button Info dialog box, click the "Link To" button, and then use *HyperCard's* built-in navigation tools to take me to the card I wished to link to the button. The link had already been made when I returned to the script, and I only had to enter minor changes (if any) to make the script do everything that I wanted it to do.

For the sake of clarity in this article, I changed several scripts, adding names in place of the ID numbers that *HyperCard* generates when using the "Link To" option. Using this automated programming tool of *HyperCard*, provided by the "Link To" option, made it easy to create most of the links. Thus, I was free to work on the important aspect of the project—ensuring that the information was accurate. I spent a minimum amount of time trying to learn a new computer language, and a maximum amount of time creating my application.

A Simple to Use, Powerful Teacher's Aid

Class Stacks for Educators

by David G. Brader

Teachers, this article is for you! If you have been seeking a way to maintain your class records in a more flexible, reliable fashion, seek no more, *HyperCard* is here!

The four *Class Stacks* described completely in the following pages will help you generate a dynamic seating chart with automatic attendance and grade posting, and provide you with an easy-access general student records system. All the information is linked and organized by class, where a class is defined by the grade (sixth grade for example), subject (English, math, history, etc.), the period during the day, and the room number.

The main stack is called "Classes." An example card from this stack is shown in Figure 1. There are several features to point out regarding this card.

The main concept to grasp right away is that the rectangles containing names (or the word empty) represent movable chairs in a classroom. So Figure 1 illustrates a seating chart for Mrs. Jones's sixth grade English class that meets fourth period in room 117. Note that the lower-right corner of the diagram shows a rectangle (labeled empty) in an area called "Chair Stack." This is a stack of empty chairs to draw from to increase the number of chairs in the classroom. Up to thirty-six seats can be accommodated in a classroom diagram. Hopefully, you have fewer students per class!

...Generate a dynamic seating chart with automatic attendance and grade posting ...

The chairs on the screen are not only used to assign seating, but also to access an individual student's record, take attendance, and post grades. They can be moved around to change the seating arrangement, or emptied and placed back in the "Chair Stack." Students may be added or deleted at any time.

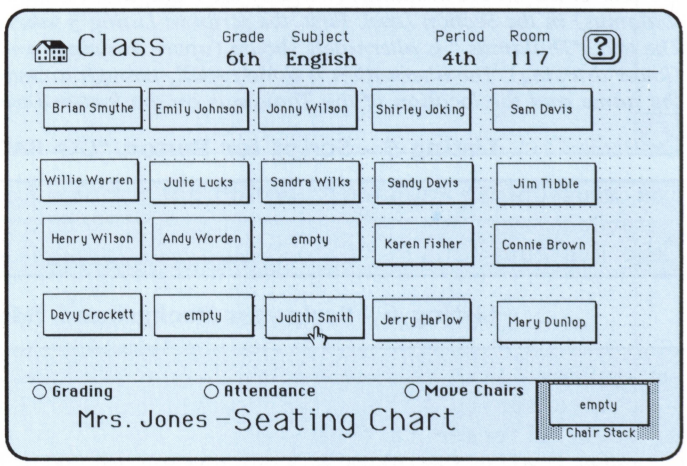


Figure 1

You can print out a classroom seating chart, such as Figure 1, and use the printed copy for taking roll or as an aid for a substitute teacher to recognize students.

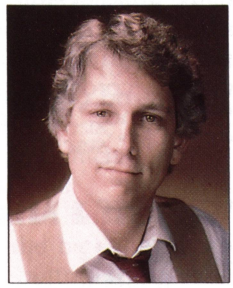
To help you understand the value of *Class Stacks*, we will examine the use of each of the four stacks as we setup Mrs. Jones's English class and then describe in detail all of the *HyperCard* objects so you can enter them into your Macintosh.

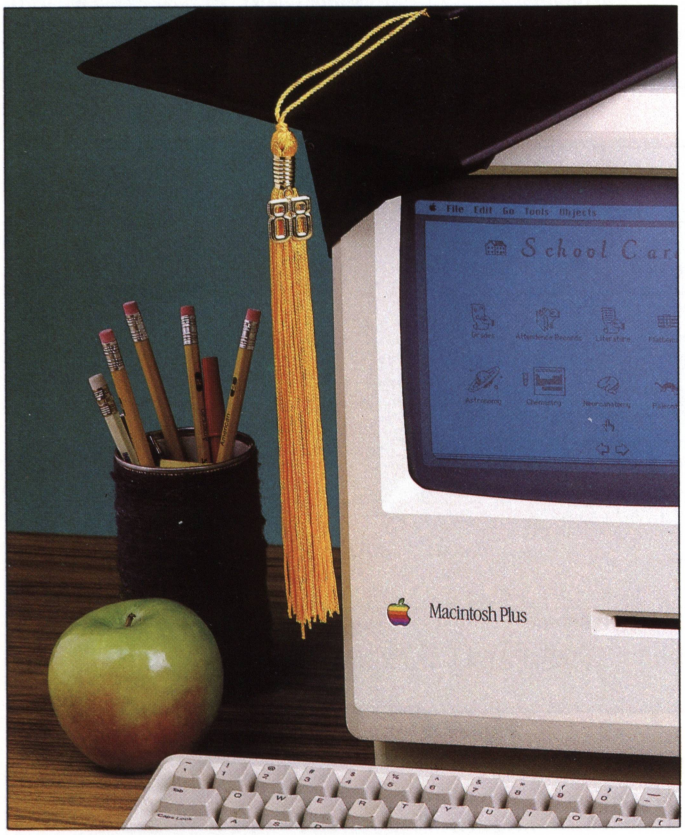
Setting Up a Classroom

Figure 2 shows the initial card that appears when you open the "Classes" stack. Do not use this card as a classroom, for this is the master template from which all other classrooms are cloned. To initiate the process of generating a new classroom, move the cursor over the words "Seating Chart" in the center bottom of the card and press the mouse button. (A transparent button is situated over these word.) The first message to appear asks, "Do you wish to build a new class?" You may answer "OK" or "Cancel" at this point. (The ability to cancel the action at this point is in case you accidentally press the hidden button during other operation.)

The second message that may appear asks, "Copy current seating plan?" This allows the teacher to copy a

David G. Brader has worked with computers for over two decades, Macintoshes since 1984, and serves as the Publisher of HyperLink Magazine.





seating chart for all the classes that are held in the same room with the same chair arrangement. Choosing "No" means the new classroom will be built from the Master template card and all chairs will be stacked in the corner as shown in Figure 2.

Dialog boxes appear next to obtain the class's grade, subject, period, classroom number, and teacher's name. After the final answer is input, the new classroom card in the "Classes" stack and its associated master backgrounds for the student, attendance, and scores records are built. These associated records are kept in the separate stacks "Students," "Attendance," and "Scores" respectively. All the stacks are linked via buttons to form a relational database.

It takes roughly three minutes to generate the new classroom because the backgrounds are cloned one object at a time from master backgrounds in the associated stacks. Just sit back and watch the action on the screen during this building process. When the process is completed, if you did not copy a current seating plan, the message shown in Figure 3 appears.

Musical Chairs

We can change modes at this point to arrange the chairs on the seating chart. Using the mouse, activate the "Move Chairs" mode button on the chart; notice

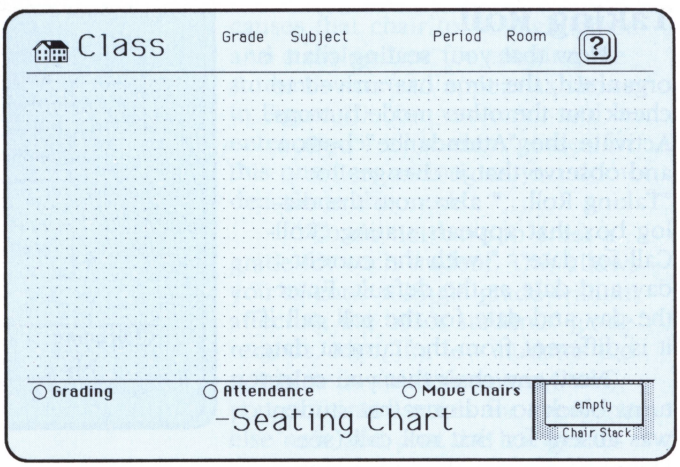


Figure 2

that it changes to "Moving..." and is highlighted. While in this mode, all you can do with a chair is drag it around the room. By (figuratively) picking up a chair from the "Chair Stack" with the mouse, you can drag it into position to represent a real chair in your classroom. This process is illustrated in Figure 4.

After positioning a chair rectangle for each real chair in your classroom, deactivate the "Moving..." mode by pressing the button again.

Assigning Seats

Now that all the chairs are positioned, it is time to associate the students' names with them and simultaneously create the students' records.

To do this is quite simple. While holding down the Command key (the cloverleaf key) momentarily press the mouse button with the cursor positioned over the chair you wish to assign.

Figure 5 shows the dialog box that appears each time you select an "empty" chair. After entering the

name and clicking on "OK," the student's name appears in the chair rectangle, and that individual student's records are created in the "Attendance," "Scores," and "Students" stacks.

What If I Misplace a Student?

If you place a student in the wrong position, switch to the "Moving..." mode and rearrange the chairs. This will not change or delete the student's records.

If the student should not be in the class, use the Command key and the mouse to select the student's chair. This time the dialog box shown in Figure 6 appears, allowing you to delete the student and the associated records in the other three stacks for that student. The chair will be marked as empty, ready for re-assignment if desired.

Taking Roll

Now that your seating chart is organized, the time has arrived to check out the other mode buttons. Activate the "Attendance" button and observe that it changes to "Taking Roll..." also note the dialog box that appears stating "Roll-Call for date - " with the current day and date as the default. Enter the day and date for the roll call if it is different from the current date.

Now any chair that you select turns black to indicate that student was absent for that roll call (see Figure 7). If you select a student by mistake, click on the same chair again to turn it back to white (which represents the student was present). After all absentees are blackened, press the "Taking Roll..." button to deactivate the "Attendance" mode and to automatically post all of the individual student records in the "Attendance" stack.

Each student's attendance record receives a dated entry line containing either "Present" or "ABSENT" depending on whether the student's chair was highlighted (blackened).

Figure 8 shows an individual student's attendance record after three roll calls. Note that you can add notes in the scrolling field such as the one shown after the second roll call entry.

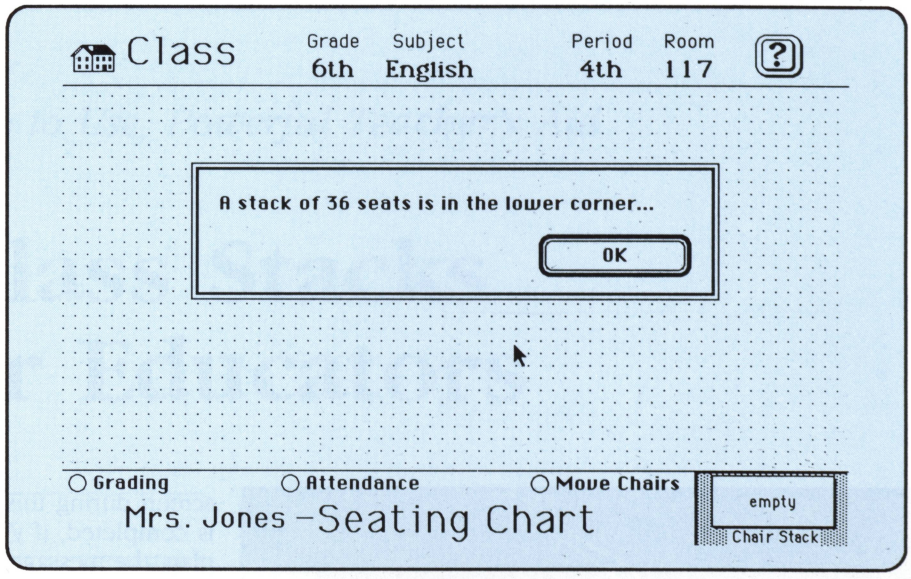


Figure 3

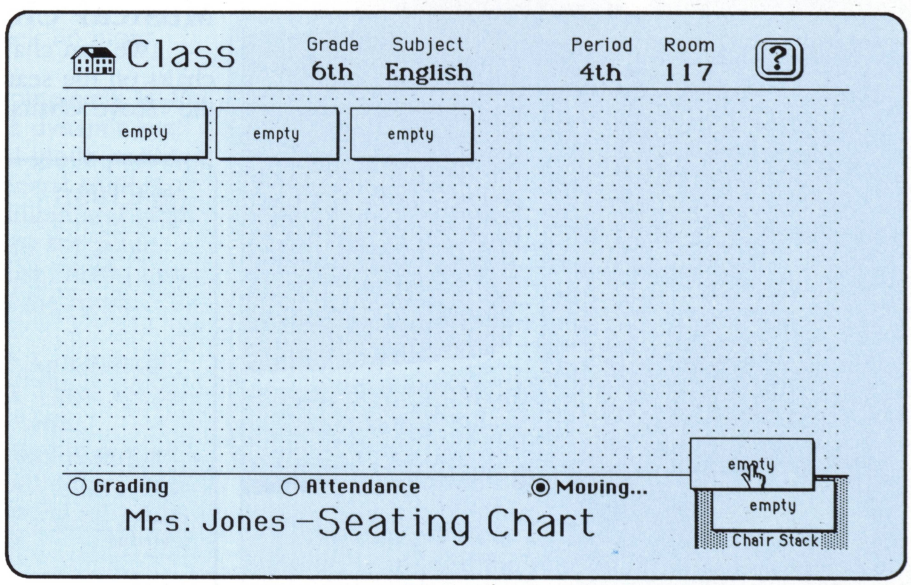


Figure 4

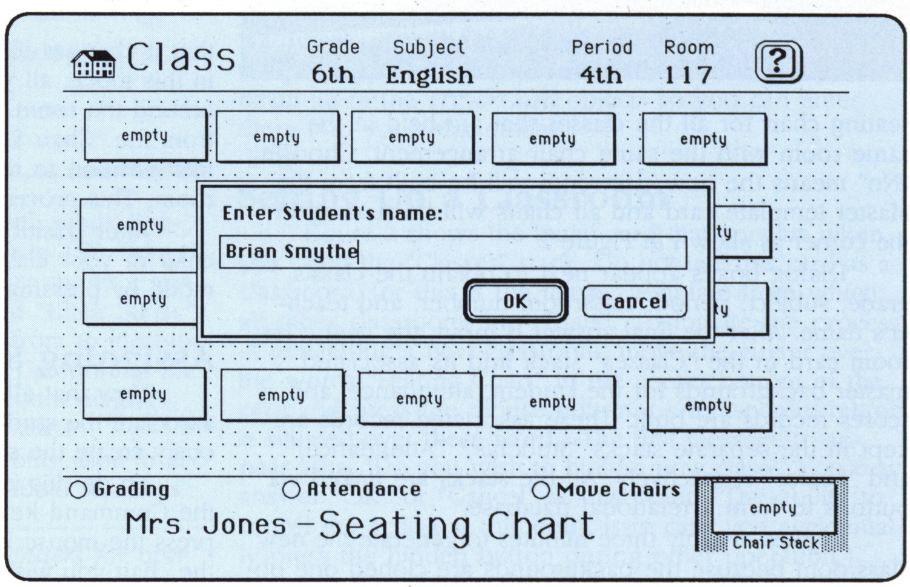


Figure 5

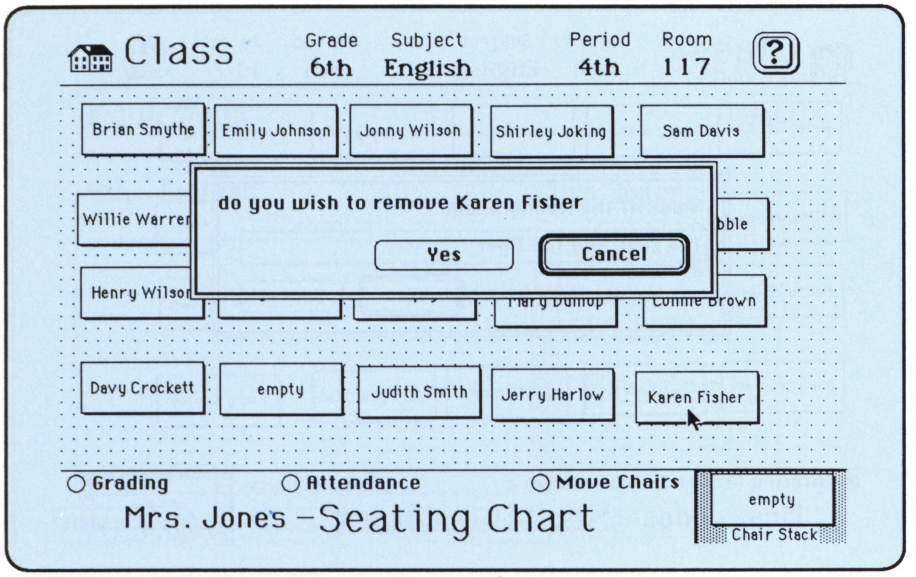


Figure 6

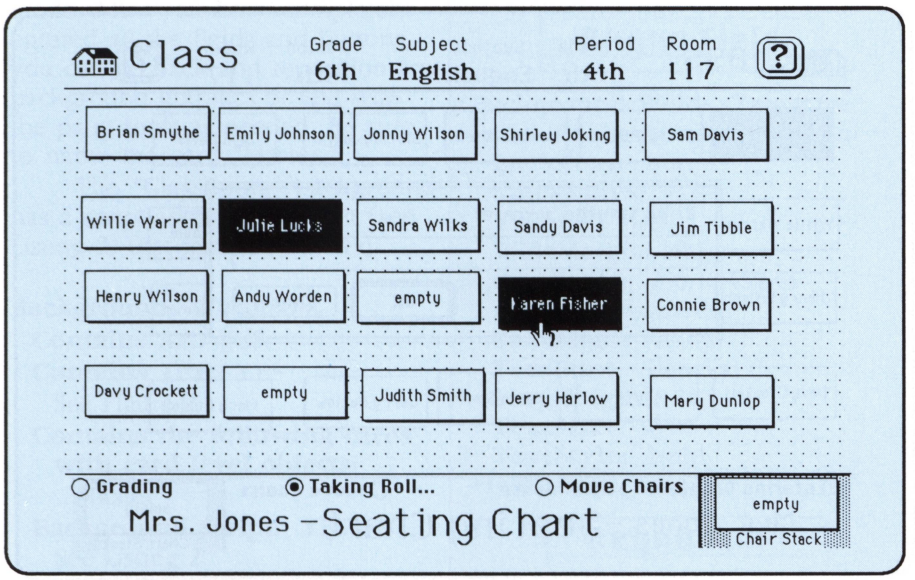


Figure 7

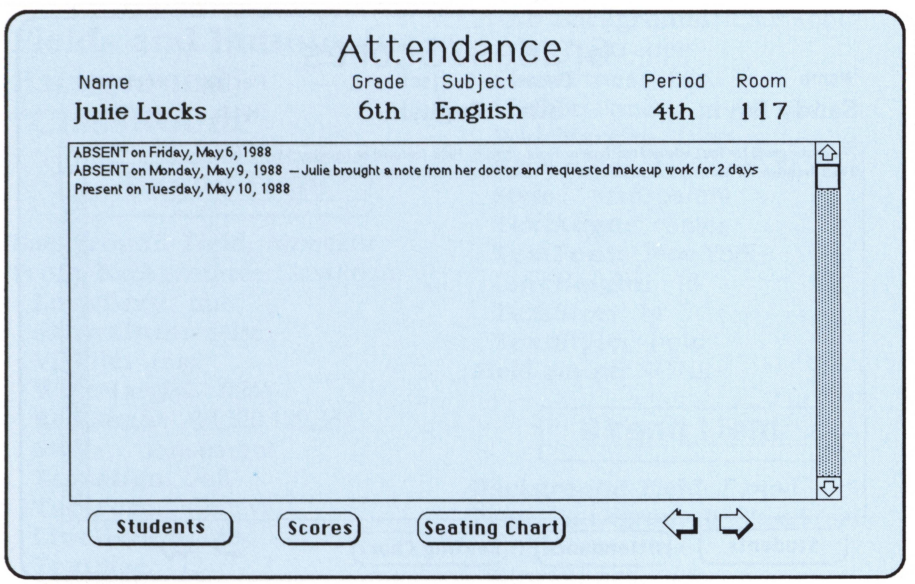


Figure 8

Making the Grades

The "Grading" mode button on the "Seating Chart" card of the "Classes" stack is used to post scores for homework, tests, reports, or any other item on which you wish to grade a student.

Activation of this button causes it to change to the highlighted "Entering Grades" button and a dialog box to appear for entering the "grade date - ." Today's day and date is the default. After entering the grade date, a second dialog box appears asking for the name of the item graded as shown in Figure 9. Enter the item for which you are entering the grades or scores. Once you have entered the item, you can enter grades. Note that you may use any method of scoring (points, percentages, or letter grades) as long as it is consistent.

In the "Entering Grades..." mode selection of a chair rectangle, that has a student assigned to it causes that chair to be highlighted and a dialog box for entering the student's score to appear as shown in Figure 10. You can enter and reenter any or all student grades for this one item as long as you do not deactivate the mode.

You do not have to enter a grade for each student. This allows you to enter a single grade without affecting the other students' grade records. Also, if a few students did not turn in an assignment (late?), you can enter scores for everyone else now and post the late grades some other day.

If a student's score is set to "none," no entry will be made in that student's grade record. So, if you make a mistake and do not wish to post a score, reselect the student's chair and replace the score with the word "none" before you post the grades.

When you are satisfied that all scores have been entered correctly, deactivate the "Entering Grades..." mode by pressing the button. All entered scores/grades will be posted to each student's grade record in the "Scores" stack.

Once a grade is posted to a student's "Grades/Scores" card in the "Scores" stack (see Figure 11), it cannot be changed because the scrolling field is locked. After posting, the only way to change a

grade/score is to make a correcting entry using the normal grade entry mode process just described.

Notice the grade/score is the first item of each line of the student's "Grades/Scores" card entry. This was done to facilitate the generation of end-of-term grade reports (by those of you that are so inclined).

Class Navigation

After opening the main stack, "Classes," use your Mac's right and left arrow keys to move to the class "Seating Chart" desired.

If no special mode is activated ("Grading," "Attendance," or "Move Chairs"), selecting a student's chair takes you to that student's record. For instance, selecting Judith Smith's chair on the "Seating Chart" (see Figure 1) takes us to her record card (see Figure 12) in the "Students" stack. Using the navigation buttons at the bottom of this card and the associated "Attendance" and "Scores" stack cards, you can move between all three stacks or go back to the "Seating Chart" in the "Classes" stack.

By the way, the "Comments" scrolling field, on the "Student" card, can be used for any notes regarding the student.

If you forget how to use these stacks and you have purchased the *StackSolutions* disk for this issue, the question mark button on the "Seating Chart" card brings up a help screen. This help button and its supporting help card is not included here in the printed version to save space.

The following pages contain all the information needed to enter all four of the *Class Stacks* ("Classes," "Students," "Attendance," and "Scores") into your Macintosh. So, warm up your Mac and let's get started...

Classes Stack

Create a new stack and name it "Classes." Open the stack info box and select the script button. Enter the script from Listing 1.

Create a background in this stack and name it "ClassRoom." Use Figure 2 as a guide for drawing the background text and graphics using the paint tools in background

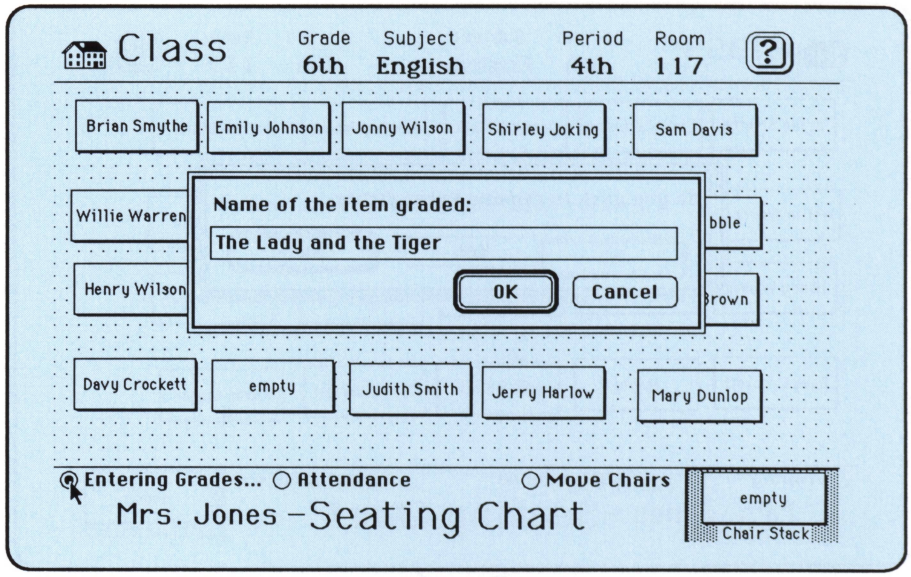


Figure 9

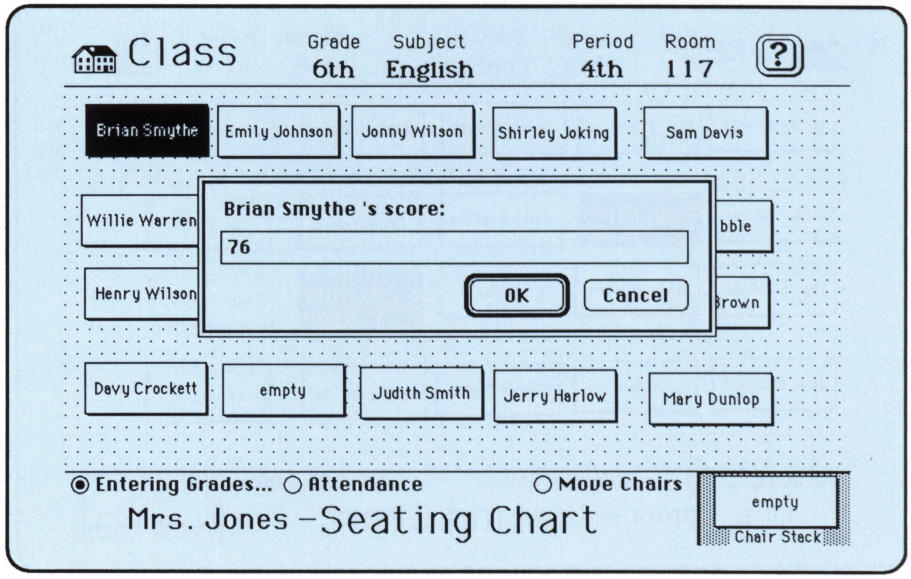


Figure 10

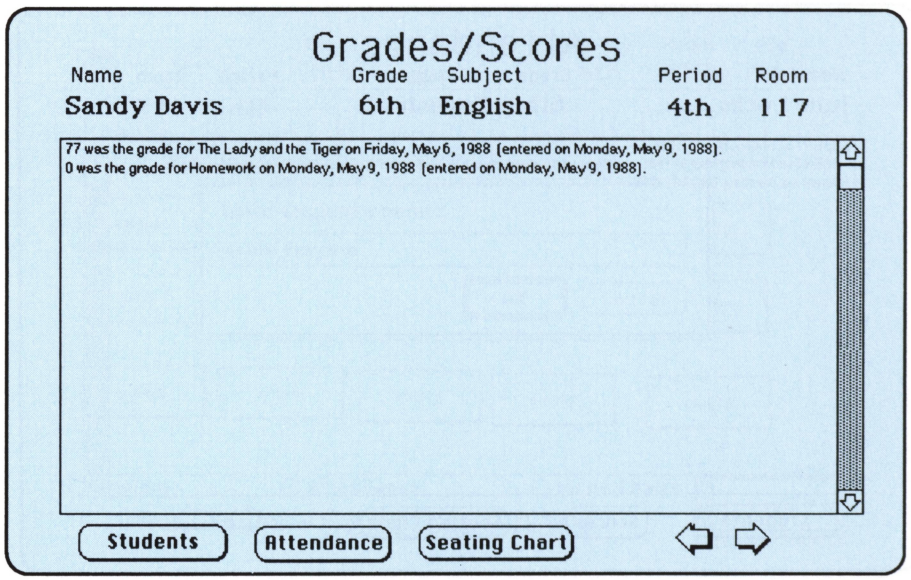


Figure 11

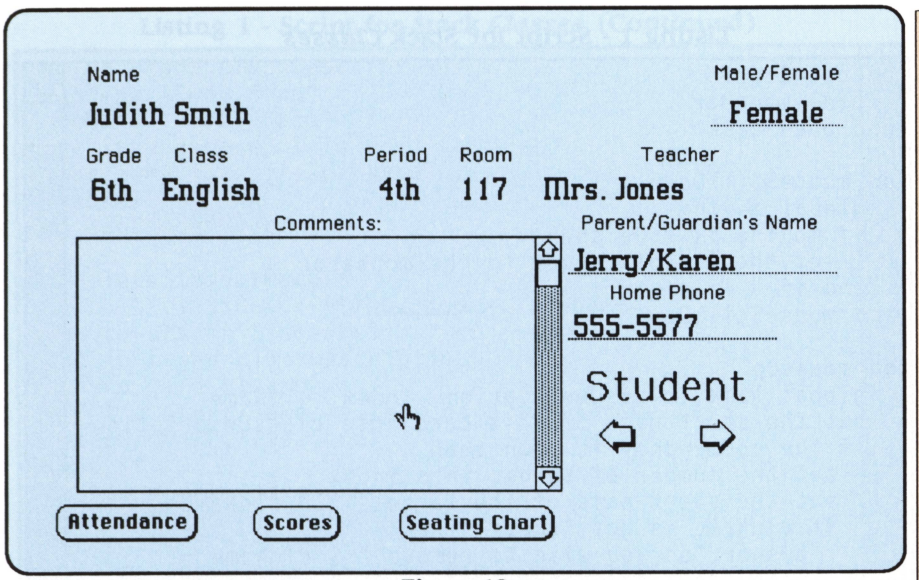


Figure 12

mode. After you have completely entered all the fields and buttons, you can go back and reposition any background graphics or text with the paint tools as needed. Be sure to name the card "Master."

The "ClassRoom" background has a script associated with it (see Listing 2) that must be entered.

Background: *ClassRoom*
Contains Art: YES
Contains Text: YES
See Figure 2
Contains the following cards with card level objects:
Master
Background Script: YES
See Listing 2

Fields and Buttons for Background "ClassRoom"

Bkgnd Field

Background Field: *NameList*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 30,320,129,332
Style: transparent
TextAlign: left
TextFont: Geneva
LineHeight: 16
TextSize: 12
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *ClassName*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: false
WideMargin: false
Rectangle: 156,16,464,43
Style: transparent
TextAlign: center
TextFont: Geneva
LineHeight: 24
TextSize: 18
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Room*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 385,21,430,42
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Period*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 331,21,373,42
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Grade*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 157,21,197,42
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Subject*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 207,21,329,42
Style: transparent
TextAlign: left
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Teacher*
From Background: ClassRoom
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 0,312,151,339
Style: opaque
TextAlign: right
TextFont: Geneva
LineHeight: 24
TextSize: 18
TextStyle: bold
Field Script: NONE

Bkgnd Button

Background Button:
Generate New Class
From Background: ClassRoom
AutoHilite: false
ShowName: false
Visible: true
Icon: None
Rectangle: 165,313,348,341
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES
See Listing 3

Bkgnd Button

Background Button: *Grading*
From Background: ClassRoom
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 1,292,141,312
Style: radioButton
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES
See Listing 4

Bkgnd Button

Background Button: *Attendance*
From Background: ClassRoom
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 140,292,303,312
Style: radioButton
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES
See Listing 5

Listing 1 - Script for Stack *Classes*

```
on openStack
  hide menuBar
end openStack

on mouseStillDown
  global moving
  if moving is true then
    set the loc of target to the mouseLoc
  end if
end mouseStillDown

on mouseUp
  global Name, classDescription, index, oldName
  put the short name of this card into classDescription
  if the commandKey is down then
    put the number of target into index
    put the short name of the target into oldName
    if oldName is not "empty" then
      answer "do you wish to remove " & oldName ¬
      with "Yes" or "Cancel"
      if it is "Yes" then
        set the name of target to "empty"
        put "empty" into line index of field "NameList"
        deleteStudentRecord
        go to stack "Classes"
        go to card classDescription
      else
        exit mouseUp
      end if
    else
      ask "Enter Student's name: "
      if it is "" then exit mouseUp
      put it into Name
      set the name of the target to Name
      put Name into line index of field "NameList"
      addStudentRecord
    end if
  else
    get the short name of the target
    put it into Name
    get the short name of this card
    put it into classDescription
    go to stack "Students"
    go to bkgnd classDescription
    go to card Name
    if the short name of this card is not Name then
      answer "No record found for " & Name
      go to stack "Classes"
      go to card classDescription
    end if
  end if
end mouseUp

on addStudentRecord
  global Name, classDescription, index
  go to stack "Attendance"
  go to bkgnd classDescription
  doMenu "Copy Card"
  doMenu "Paste Card"
  set the name of this card to Name
  put Name into field "Name"
  go to stack "Scores"
  go to bkgnd classDescription
  doMenu "Copy Card"
  doMenu "Paste Card"
  set the name of this card to Name
  put Name into field "Name"
  go to stack "Students"
```

Listing 1 - Script for Stack *Classes* (Continued)

```
 go to bkgnd classDescription
  doMenu "Copy Card"
  doMenu "Paste Card"
  set the name of this card to Name
  put Name into field "Name"
  tabKey
end addStudentRecord

on deleteStudentRecord
  global classDescription, index, oldName
  go to stack "Students"
  go to bkgnd classDescription
  go to card oldName
  if the short name of this card is oldName then
    doMenu "Delete Card"
  end if
  go to stack "Attendance"
  go to bkgnd classDescription
  go to card oldName
  if the short name of this card is oldName then
    doMenu "Delete Card"
  end if
  go to stack "Scores"
  go to bkgnd classDescription
  go to card oldName
  if the short name of this card is oldName then
    doMenu "Delete Card"
  end if
end deleteStudentRecord
```

Listing 2 - Script for Background *ClassRoom*

```
on mouseUp
  global grading, scoreList, takingAttendance, ¬
  rollCall, moving
  get the short name of target
  put it into Name
  get the number of target
  put it into index
  if grading is true then
    if Name is "empty" then exit mouseUp
    set the hilite of target to true
    put item index of scoreList into score
    ask Name && "'s score:" with score
    put it into item index of scoreList
    if it is "none" then
      set the hilite of target to false
    end if
  else
    if takingAttendance is true then
      if Name is "empty" then exit mouseUp
      if the hilite of target is false then
        set the hilite of target to true
        put "ABSENT" into item index of rollCall
      else
        set the hilite of target to false
        put "Present" into item index of rollCall
      end if
    else
      if moving is true then
        exit mouseUp
      else
        pass mouseUp
      end if
    end if
  end if
end mouseUp
```

Bkgnd Button

Background Button: *Home*
From Background: ClassRoom
AutoHilite: false
ShowName: false
Visible: true
Icon: 1011
Rectangle: 2,10,39,38
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseStillDown
  exit mouseStillDown
end mouseStillDown

on mouseUp
  go Home
end mouseUp
```

Bkgnd Button

Background Button: *Move Chairs*
From Background: ClassRoom
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 303,292,406,312
Style: radioButton
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES
See Listing 6

Card Button

Card Button: *empty*
From Card: Master
From Background: ClassRoom
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 423,296,504,331
Style: shadow
TextAlign: center
TextFont: Geneva
LineHeight: 13
TextSize: 10
TextStyle: plain
Button Script: NONE

Listing 3 - Script for Background Button *Generate New Class*

```
on mouseStillDown
  exit mouseStillDown
end mouseStillDown

on mouseUp
  global grading, takingAttendance, ¬
  moving, NumOfButtons, NumOfFields, ¬
  classDescription
  if grading is true or moving is true ¬
  or takingAttendance is true then
    exit mouseUp
  else
    beep
    beep
    answer ¬
    "Do you wish to build a new class?"¬
    with "OK" or "Cancel"
    if it is "Cancel" then exit mouseUp
    if the short name of this card ¬
    is not "Master" then
      answer ¬
      "Copy current seating plan?" ¬
      with "Yes" or "No"
      put it into plan
    else
      put "No" into plan
    end if
    if plan is "No" then
      go to first card
      doMenu "Copy Card"
      doMenu "Paste Card"
      choose button tool
      click at ¬
      the location of card button 1
      doMenu "Copy Button"
      repeat with x = 1 to 35
        domenu "Paste Button"
      end repeat
      choose browse tool
    else
      doMenu "Copy Card"
      doMenu "Paste Card"
      repeat with x = 1 to 36
        set the name ¬
        of card button x to "empty"
      end repeat
    end if
    ask "What is the class grade?"
    put it into grade
    ask "What is the class subject?"
    put it into className
    ask "What is the class period?"
    put it into period
    ask "What is the class room?"
    put it into room
    ask "What is Teacher's name?"
    put it into teacher
    put grade & period & room & ¬
    className into classDescription
    put grade into field "Grade"
    put className into field "Subject"
    put period into field "Period"
    put room into field "Room"
    put teacher into field "Teacher"
    repeat with x = 1 to 40
      put "empty" into ¬
      line x of field "NameList"
    end repeat
    set the name of ¬
    this Card to classDescription
    push card
    go to stack "Students"
    put 5 into NumOfButtons
    put 10 into NumOfFields
    buildBackground
    put grade into field "Grade"
    put className into field "ClassName"
    put period into field "Period"
    put room into field "Room"
    put teacher into field "Teacher"
    go to stack "Attendance"
    put 5 into NumOfButtons
    put 6 into NumOfFields
    buildBackground
    put grade into field "Grade"
    put className into field "ClassName"
    put period into field "Period"
    put room into field "Room"
    go to stack "Scores"
    put 5 into NumOfButtons
    put 6 into NumOfFields
    buildBackground
    put grade into field "Grade"
    put className into field "ClassName"
    put period into field "Period"
    put room into field "Room"
    pop card
    if plan is "No" then
      answer "A stack of 36 seats ¬
      is in the lower corner..."
    end if
  end if
end mouseUp

on buildBackground
  global NumOfButtons, NumOfFields, ¬
  classDescription
  doMenu "New Background"
  set the name of this background to ¬
  classDescription
  push card
  go to next card
  doMenu "Background"
  choose select tool
  doMenu "Select All"
  doMenu "Copy Picture"
  pop card
  doMenu "Background"
  doMenu "Paste Picture"
  repeat with x = 1 to NumOfFields
    push card
    go to next card
    doMenu "Background"
    choose field tool
    click at loc of bkgnd field x
    doMenu "Copy Field"
    pop card
    doMenu "Background"
    doMenu "Paste Field"
  end repeat
  repeat with x = 1 to NumOfButtons
    push card
    go to next card
    doMenu "Background"
```

Listing 3 - *Generate New Class* (Continued)

```
    choose button tool
     click at loc of bkgnd button x
     doMenu "Copy Button"
     pop card
     doMenu "Background"
     doMenu "Paste Button"
   end repeat
   choose browse tool
end buildBackground
```

Listing 4 - Script for Bkgnd Button *Grading*

```
on mouseStillDown
  exit mouseStillDown
end mouseStillDown

on mouseUp
  global grading, scoreList, element, ¬
  takingAttendance, moving, ¬
  classDescription, gradeDate, postDate
  put the short name of this card ¬
  into classDescription
  if takingAttendance is true ¬
  or moving is true then
    set hilite of target to false
  else
    if hilite of target is false then
      set hilite of target to true
      put true into grading
      set the name of target ¬
      to "Entering Grades..."
      ask "grade date -" with ¬
      the long date
      put it into gradeDate
      put the long date into postDate
      repeat with x = 1 to 40
        put "none" into item x ¬
        of scoreList
      end repeat
      ask "Name of the item graded:"
      put it into element
    else
      set the name of target ¬
      to "Posting Grades..."
      postGrades
      set the name of target ¬
      to "Grading"
      set hilite of target to false
      put false into grading
    end if
  end if
end mouseUp

on postGrades
  global grading, scoreList, element, ¬
  takingAttendance, moving, ¬
  classDescription, gradeDate, postDate
  put field "NameList" into nameList
  repeat with x= 1 to 40
    if line x of nameList ¬
    is not "empty" then
      set the hilite of card button x ¬
      to false
    end if
  end repeat
  go to stack "Scores"
  go to bkgnd classDescription
```

Listing 4 - *Grading* (Continued)

```
  repeat with x= 1 to 40
    if line x of nameList ¬
    is not "empty" and item x of ¬
    scoreList is not "none" then
      go to card line x of nameList
      put  item x of scoreList & ¬
      " was the grade for " & element ¬
      & " on " & gradeDate & ¬
      "  (entered on " & postDate & ¬
      ")." & return after field "Scores"
    end if
  end repeat
  go to stack "Classes"
  go to card classDescription
end postGrades
```

Listing 5 - Script for Bkgnd Button *Attendance*

```
on mouseStillDown
  exit mouseStillDown
end mouseStillDown

on mouseUp
  global grading, rollCall, ¬
  takingAttendance, moving, ¬
  classDescription, rollCallDate
  put the short name of this card ¬
  into classDescription
  if grading is true ¬
  or moving is true then
    set hilite of target to false
  else
    if hilite of target is false then
      set hilite of target to true
      put true into takingAttendance
      set the name of target ¬
      to "Taking Roll..."
      ask "RollCall for date -" with ¬
      the long date
      put it into rollCallDate
      repeat with x = 1 to 40
        put "Present" into item x ¬
        of rollCall
      end repeat
    else
      set the name of target ¬
      to "Posting Attendance..."
      postAttendance
      set the name of target ¬
      to "Attendance"
      set hilite of target to false
      put false into takingAttendance
    end if
  end if
end mouseUp

on postAttendance
  global rollCall, classDescription, ¬
  rollCallDate
  put field "NameList" into nameList
  repeat with x= 1 to 40
    if line x of nameList ¬
    is not "empty" then
      set the hilite of card button x ¬
      to false
    end if
  end repeat
```

Students Stack

Create a new stack and name it "Students." Open the stack info box and select the script button. Enter the following script:

```
on mouseUp
  get the short name ¬
  of this card
  put it into Name
  get the short name ¬
  of this bkgnd
  put it into ¬
  classDescription
  go to stack the short ¬
  name of the target
  go to bkgnd ¬
  classDescription
  go to card Name
end mouseUp
```

The above script is activated by clicking the mouse button on any object that doesn't have a mouse button handler. By naming a button the precise name of a destination stack, leaving out any script for that button, and using this stack level script:

```
go to stack the short ¬
name of the target
```

we have produced a neat way of navigating between stacks.

The rest of the script in this stack-level handler forms the relative linking logic to make sure we get to the proper student's information within the right class of the destination stack.

This stack handler is in the "Attendance" and "Scores" stacks too. It works with the "Students," "Scores," and "Attendance" background buttons within the three stacks to navigate between them.

Create a background in this "Students" stack and name it "Master." Use Figure 12 as a guide for drawing the background text using the paint tools in background mode.

After you have completely entered all the fields and buttons you can go back and reposition any background graphics or text with the paint tools as needed.

Be sure to also name the card "Master." The "Master" background has no script associated with it. Neither is there any card level script.

Listing 5 - Script for Bkgnd Button *Attendance* (Continued)

```
  go to stack "Attendance"
  go to bkgnd classDescription
  repeat with x= 1 to 40
    if line x of nameList is not "empty" then
      go to card line x of nameList
      put item x of rollCall & " on " & rollCallDate & ¬
      return after field "Attendance"
    end if
  end repeat
  go to stack "Classes"
  go to card classDescription
end postAttendance
```

Listing 6 - Script for Background Button *Move Chairs*

```
on mouseStillDown
  exit mouseStillDown
end mouseStillDown

on mouseUp
  global grading, nameList, rollCall, takingAttendance,
moving
  if grading is true or takingAttendance is true then
    set hilite of target to false
  else
    if hilite of target is false then
      set hilite of target to true
      put true into moving
      set the name of target to "Moving..."
    else
      set hilite of target to false
      put false into moving
      set the name of target to "Move Chairs"
    end if
  end if
end mouseUp
```

Background: *Master*
Contains Art: NO
Contains Text: YES
See Figure 12
Contains the following cards with card level objects:
none
Background Script: NO

Fields and Buttons for Background "Master"

Bkgnd Field

Background Field: *Name*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 31,54,419,82
Style: transparent
TextAlign: left
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Sex*
From Background: Master
LockText: false
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 424,54,510,82
Style: transparent
TextAlign: center
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Grade*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 30,102,73,128
Style: transparent
TextAlign: center
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *ClassName*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 78,102,205,128
Style: transparent
TextAlign: left
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Period*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 210,102,251,128
Style: transparent
TextAlign: center
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Room*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 257,102,310,128
Style: transparent
TextAlign: center
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Teacher*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 316,102,505,128
Style: transparent
TextAlign: left
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Parent*
From Background: Master
LockText: false
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 335,147,507,173
Style: transparent
TextAlign: left
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *PhoneNumber*
From Background: Master
LockText: false
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 335,188,469,215
Style: transparent
TextAlign: left
TextFont: Athens
LineHeight: 24
TextSize: 18
TextStyle: plain
Field Script: NONE

Bkgnd Field

Background Field: *Comments*
From Background: Master
LockText: false
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 29,144,332,304
Style: scrolling
TextAlign: left
TextFont: Helvetica
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: NONE

Bkgnd Button

Background Button: *Left Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 9301
Rectangle: 350,256,381,281
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to prev card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button: *Right Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 27009
Rectangle: 415,255,444,282
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to next card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button:
Seating Chart
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 230,313,327,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  get the short name ¬
  of this bkgnd
  put it into ¬
  classDescription
  go to stack "Classes"
  go to card ¬
  classDescription
end mouseUp
```

Bkgnd Button

Background Button: *Attendance*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 16,312,104,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Bkgnd Button

Background Button: *Scores*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 139,313,193,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Attendance Stack

Create a new stack and name it "Attendance." Open the stack info box and select the script button. Enter the following script:

```
on mouseUp
  get the short name ¬
  of this card
  put it into Name
  get the short name ¬
  of this bkgnd
  put it into ¬
  classDescription
  go to stack the short ¬
  name of the target
  go to bkgnd ¬
  classDescription
  go to card Name
end mouseUp
```

Create a background and name it "Master." Use Figure 8 as a guide for drawing the background text using the paint tools in the background mode. After you have entered all the fields and buttons, go back and reposition any background text as needed.

Be sure to also name the card "Master." The "Master" background has no script associated with it. Neither is there any card level script.

Background: *Master*
Contains Art: NO
Contains Text: YES
See Figure 8
Contains the following cards with card level objects:
none
Background Script: NO

Fields and Buttons for Background "Master"

Bkgnd Field

Background Field: *Name*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 4,41,183,66
Style: transparent
TextAlign: left
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Grade*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 186,41,228,65
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *ClassName*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 239,41,365,65
Style: transparent
TextAlign: left
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Period*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 378,42,419,66
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Room*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 431,42,483,66
Style: transparent

TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Attendance*
From Background: Master
LockText: false
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 4,71,507,306
Style: scrolling
TextAlign: left
TextFont: Helvetica
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: NONE

Bkgnd Button

Background Button: *Left Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 9301
Rectangle: 385,308,416,333
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to prev card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button: *Right Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 27009
Rectangle: 423,307,452,334
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to next card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button: *Seating Chart*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 230,313,327,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  get the short name ¬
  of this bkgnd
  put it into ¬
  classDescription
  go to stack "Classes"
  go to card ¬
  classDescription
end mouseUp
```

Bkgnd Button

Background Button: *Scores*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 139,313,193,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Bkgnd Button

Background Button: *Students*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 16,312,110,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Scores Stack

Create a new stack and name it "Scores." Open the stack info box and select the script button. Enter the same script listing that is used for the "Attendance" stack.

Create a background and name it "Master." Use Figure 11 as a guide for the background text.

Be sure to also name the card "Master." There are no background or card level scripts.

Background: *Master*
Contains Art: NO
Contains Text: YES
See Figure 11
Contains the following cards with card level objects:
none
Background Script: NO

Fields and Buttons for Background "Master"

Bkgnd Field

Background Field: *Name*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 4,41,183,66
Style: transparent
TextAlign: left
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Grade*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 186,41,228,65
Style: transparent
TextAlign: center

TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *ClassName*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 239,41,365,65
Style: transparent
TextAlign: left
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Period*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 378,42,419,66
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Room*
From Background: Master
LockText: true
ShowLines: false
Visible: true
WideMargin: false
Rectangle: 431,42,483,66
Style: transparent
TextAlign: center
TextFont: New York
LineHeight: 18
TextSize: 14
TextStyle: bold
Field Script: NONE

Bkgnd Field

Background Field: *Scores*
From Background: Master
LockText: true
ShowLines: true
Visible: true
WideMargin: false
Rectangle: 4,71,507,306
Style: scrolling
TextAlign: left
TextFont: Helvetica
LineHeight: 12
TextSize: 9
TextStyle: plain
Field Script: NONE

Bkgnd Button

Background Button: *Left Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 9301
Rectangle: 385,308,416,333
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to prev card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button: *Right Arrow*
From Background: Master
AutoHilite: false
ShowName: false
Visible: true
Icon: 27009
Rectangle: 423,307,452,334
Style: transparent
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  go to next card ¬
  in this bkgnd
end mouseUp
```

Bkgnd Button

Background Button: *Seating Chart*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 230,313,327,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: YES

```
on mouseUp
  get the short name ¬
  of this bkgnd
  put it into ¬
  classDescription
  go to stack "Classes"
  go to card ¬
  classDescription
end mouseUp
```

Bkgnd Button

Background Button: *Attendance*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 139,313,193,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Bkgnd Button

Background Button: *Students*
From Background: Master
AutoHilite: false
ShowName: true
Visible: true
Icon: None
Rectangle: 16,312,110,335
Style: roundRect
TextAlign: center
TextFont: Chicago
LineHeight: 16
TextSize: 12
TextStyle: plain
Button Script: NO

Note that grading and attendance reports can be designed using a product like *HyperCONTROL* from Nordic Software or *Reports* from Activision making *Class Stacks* a real class act.

HyperCard™

A VIDEO TUTORIAL

Your personal instructor on video tape

Loaded with easy to follow shots of the screen

Plenty of entertaining graphic examples

1 hour 40 minutes of well paced instruction

Professionally written, produced, and tested

How To Create
- **Cards**
- **Stacks**
- **Buttons**
- **Scripts**

This video tape is designed for many of us who are absolutely brilliant, creative and capable of mastering almost anything in life. Our only downfall is following an instruction manual. After we purchase a new piece of computer software then attempt to follow the written tutorial, we become impatient and often confused. Ultimately, we wind up imposing on a knowledgable friend to help us out. After a brief session, we're confident that our problems are solved; that is until about 11:00 that evening when we get hopelessly stuck. We place a call to our friend, the expert, to ask just one more seemingly intelligent question about the program. Guilt ridden after the late night call, we never ask for help again.

This video tape was designed for us. The program is a step by step visual walk through the basics of HyperCard™. The pace is comfortable, and the style entertaining.

The tape is perfect for business executives, students, and anyone else that would like a comfortable introduction to the usage and capabilities of this terrific new program from Apple® Computer, Inc.

Name _____

Street _____

City _____ State _____ Zip _____

Telephone Number ( )_____-_____

Please Charge on ☐ MC ☐ Visa ☐ Am Express ☐ Check Enclosed

Account # _____ Expires _____

Signature _____

Mail to: Voice & Video, Inc. 5038 Ruffner Street San Diego, Ca. 92111

☐ Please send details on Advanced HyperCard™ tape

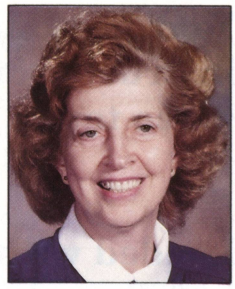
$49.95

☐ VHS ☐ Beta ☐ 8mm

HyperCard™ Video Tutorial	$49.95
Shipping and Handling	$ 3.00
In CA. add 6% Sales Tax ($3.00)	_____
Total	$ _____

Money Back Guarantee

Short Stacks



In Praise of HyperCard's Easy Access to New Users

by Joan Donaldson

About mid-October I began thinking. . . . Yes, that simple BASIC grades program I wrote about four years ago did cause students to feel some healthy competition, and security in knowing their standing in the class, but it didn't always provide me with all the information I needed. There were still a few students who remained at a level of learning which was discouraging to them, and to me. Hurriedly written "anecdotal" notes about those students were scribbled down whenever I could find time, and put wherever I could put them into the computer. I felt good about keeping the information, but panicky about finding it when I needed it for reference for talks with the students.

How could I organize this information without purchasing some new program? Having made my diary with *HyperCard*, I decided to face this more complicated task. I would make a place (I like that better than the word "program") where I could organize my notes about students. A place where I could easily find the list of students and then go to any of the students I wished to write about. I call it my "Anecdotal Records" stack.

Counseling Helper

I now can go to these records for pertinent information when I wish to let a counselor know the student is failing or needs some other type of help. My confidential notes of this kind dropped into the counselor's mail box save time for both the counselor and me. I also review these notes before I have a parent conference. As a teacher who sees 150 students daily, I simply am not able to instantly recall at a minute's notice a specific detail about one student, but I can easily go to my *HyperCard* "place," read my notes, and excerpt portions of them for my conference notes. I believe the way I have used this stack for my seniors is also well suited for a college class. I typed in only the names of those students who evidenced some problems at the end of four weeks. I knew it was important to make sure they were aware of the possibility of failing English, because this failure could keep them from graduating.

I also typed in all of the names for each Freshmen class (a new stack for each class), as the cultural shock of coming to high school and finding different expectations on the part of teachers and principals causes students to flounder, and almost founder . . . oops, couldn't resist that. Notes kept about these students are quite a bit different from those kept about seniors and college students, but this stack meets this notekeeping need effectively as well, as I can go back and forth between the index (see Figure 1) and individual student card (see Figure 2). I can just put in the notes about any particular student at the end of the school day. I can also copy this whole stack, rename it, and have another place for a different set of student notes.

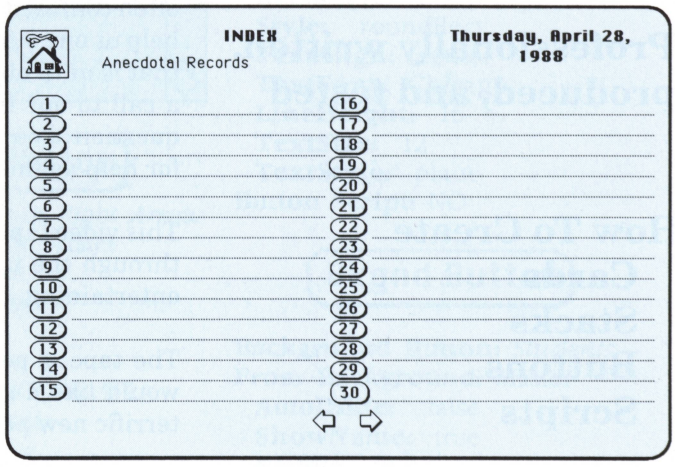


Figure 1

I especially like the fact that *HyperCard* stores this confidential information away from prying eyes of anyone who might pick up a notebook lying on a desk. No student can read about himself or another student. This improves a student's trust in the teacher. Such trust is a valuable ingredient in a teacher/student relationship.

Creating "Anecdotal Records"

This stack has two backgrounds, one called "Index," the other, "Student Cards." To create this stack,

Joan Donaldson has been teaching English to Bay Area students for many years in both High School and Junior College environments.

A Landmark Weekend in Hypermedia History

A Conference and Exhibition

Join the Thousands in San Francisco

Seminars for Developers & End Users
Bill Atkinson on HyperCard Version 1.2 • Ted Nelson, "Toward a Universal Hypermedia Standard" • CD Rom & Video Disk • Advanced HyperTalk Programming • Hypermedia as a Mass Medium • Custom Corporate Solutions • Expert Systems & Artificial Intellegence • Targeting Solutions for Vertical Markets • HyperText Systems • The Business of Hypermedia • HyperCard User Group Forum • Higher Education Solutions • & Much More.

What Is a StackMart? In addition to commercial exhibit booths, individual StackWare developers will for the firtst time have a chance to display their product as a full exhibitor. StackMart tables are $95, and that includes an electric plug, table, backdrop and sign. The StackMart is a critical part of the HyperExpo because it is based upon the ingenuity of the individual developer that the future of hypermedia rests. StackMart tables will be available to the day of the Expo.

One Day Conference & Exhibits
$25 by June 1, $30 at the Door
Two Day Conference & Exhibits
$40 by June 1, $50 at the Door
Contact: American Expositions, Inc. 110 Greene St., Ste. 703, NY, NY 10012 • 212-226-4141

The HyperExpo Faculty
(A Partial Listing)

- Steven All, *Apple Computer*
- Bill Atkinson, *Creator of HyperCard*
- Peter Black , *Xiphias*
- David Brader, *HyperLink*
- Stuart Brand, *Whole Earth*
- Bryan Carter, *Apple Computer*
- Harry Chesley, *Apple Computer*
- Bill Cleary, *Cleary Comm.*
- Raines Cohen, *BMUG*
- Scott Cook, *Intuit*
- Stan Cornyn, *Warner New Media*
- Jerry Daniels, *Author/Journalist*
- Fred Davis, *Editor, MacUser*
- Dave Drake, *Highlighted Data*
- Steve Enzer, *Claris*
- David Gewirtz, *Hyperpress*
- Richard Gingras, *MediaWorks*
- Danny Goodman, *"The Complete HyperCard Handbook"*
- Paul Heckel, *Quickview Systems*
- Joe Hutsko, *Claris*
- Keith Jordan, *Whole Earth*
- Carol Kaehler, *Apple Computer*
- Ted Kaehler, *Apple Computer*
- Lynn Knerr, *Apple Computer*
- Kristee Kreitman, *Apple Computer*
- David Leffler, *Apple Computer*
- Jan Lewis, *Editor, HyperAge*
- Mike Mosher, *Apple Computer*
- Ted Nelson, *Creator, Hypertext, Xanadu/Autodesk*
- Bill Nisen, *Owl International*
- Lon Poole, *Author/Journalist*
- Willi Powell, *Apple Computer*
- Craig Ragland, *Interactive Design*
- Stu Roberson, *Activision*
- Dan Ruby, *Editor, MacWeek*
- Rachel Rutherford, *Apple Computer*
- Sheldon Sax, *Middlebury College*
- Kirk Scott, *Kirk Scott Design*
- Dan Shafer, *Author, "HyperTalk Programming"*
- Bob Stein, *Voyager*
- Roger Strukhoff, *Editor, CD Rom Review*
- Scott Swartz, *Symmetry*
- Tom Tafuto, *Kabisa*
- Barbara Tien, *Farallon Computing*
- Wes Thomas, *Author*
- Tay Vaughn, *The Hypermedia Group*
- Clair Whitmer, *MacWeek*
- Roger Wood, *Editor, HyperLink*
- Paul Zuzelo, *Activision*

HyperExpo & StackMart will also be held in Boston October 15-16, 1988

HyperExpo & StackMart

San Francisco Civic Auditorium • June 11-12, 1988

select "New Stack" from the File menu. To ensure that you begin with a totally blank background be sure the "Copy Current Background" is not checked.

Next, select "Background" from the Edit menu to be sure you are working in the background so a "New Card" command will copy all the essential parts of this card if you wish to create more index cards. (In this case, if your class is larger than 30 students.) Name the stack , then go "Home" and make a new button named "Anecdotal Records" (you can always change this name later if you wish.) Open the script editor for this button and enter the following script:

```
on mouseUp
  go to stack ¬
  "Anecdotal Records"
end mouseUp
```

Use the button to go back to the stack you are creating. Select "Background" again. Make these fields in this background:

- A field named "Date."
- A field in which you may write "Index" in any font you choose.
- A field with its chosen font in which you can write the name of the class, or whatever you wish to see when working with the card.

Put the following in the Stack script:

```
on openCard
  put the long date into ¬
  background field "Date"
end openCard
```

You can now add your own "Home" button (I have one with its own Icon) in this background.

Next, select "New Button" from the Objects menu and make it the size you see in Figure 1. Name it "1" (no quotes) by double-clicking on the button and entering the name in the "Button Info" dialog box. Be sure "Show Name" is checked. Close the dialog box and place the button as I have.

Holding down the Option key, click and drag down on this button. You will thus create a second button. Move it down just below the first button to the next place in line. If you hold down the Shift key and the Option key at the same time as you drag, the buttons will remain in perfect vertical alignment because the Shift key constrains your movement to the direction you first drag—in this case vertically. After you have created all 30 buttons, change their names to corespond with their sequencing, like in Figure 1.

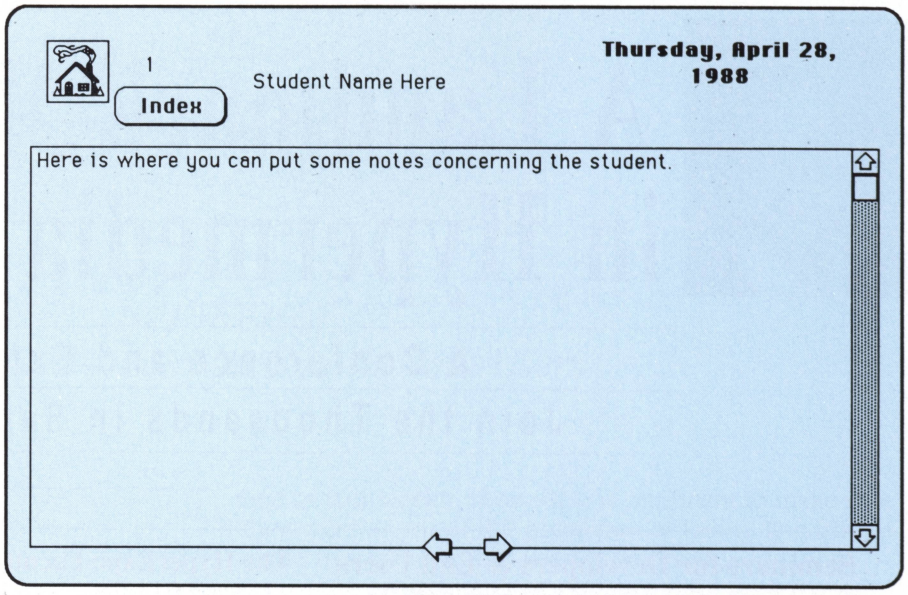


Figure 2

Create two rectangular fields, one beside each column of buttons, and click "Show Lines" within the "Field Info" dialog box so that you will have a line upon which to write. You can choose a font (10 point works well) for these fields.

For ease of moving from this background to the individual student card while I am working with it, I placed copies of the "next" and "prev" arrows at the bottom of this index card and the bottom of the individual student card. I can always remove them later.

For the Student Cards

Create a "New Background" from the Objects menu. Be sure you are in background as you will want to make several of these cards, now or later.

Create the field for the date, and be sure to name it "Date" so the stack script that we entered above can find the field. Create a field in which to place the name of the student, using a font style that you like. Create a tiny field in which to place the number corresponding to the Index number.

Each card has a scrolling field because I write a great deal about students. If you do not need this much room, just eliminate that feature. Teachers in the lower grades might want to keep the scrolling field to keep more extensive records for review with parents on the day dedicated to conferences.

Make a new background button titled "Index" (be sure "Show Name" is checked) with the following script:

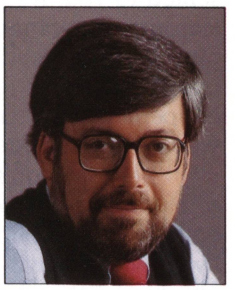
```
on mouseUp
  go to first card
end mouseUp
```

Now go back to the first card. Instead of putting a different script in each of the 30 buttons for navigation, here is a slick way to write one script for all 30 buttons, and not even place it in the button scripts! Just select "Background Info" from the Object menu, click on "Script," and put the following script in there:

```
on mouseUp
  if the first word of the ¬
  target is "Card" then ¬
  exit mouseUp
  else go to card (short ¬
  name of the target + 1)
end mouseUp
```

Here, `short name of the target` will be the name of the background button clicked on. Note that the Index card is the first card in the stack, so background button "1" is gets linked to card 2—the First Student.

An Excerpt from His New Book—
Danny Goodman's HyperCard Developer's Guide



A Different Approach to HyperTalk

by Danny Goodman

The following excerpt is taken from Danny Goodman's new release from Bantam Books—*Danny Goodman's HyperCard Developer's Guide*. The first section of this article is Chapter 12, "A Different Approach to HyperTalk." The last several pages are from Chapter 20, "Solving Searching and Sorting Mysteries." This latter section gives a good overview of all the built-in search and sort routines available with the latest release of *HyperCard*, version 1.2.

From Chapter 12

About one-half of *The Complete HyperCard Handbook* is devoted to the *HyperTalk* language. Because the book is both an introduction and ready reference to the various commands, functions, and properties of *HyperTalk* and *HyperCard*, most of the *HyperTalk* discussions are organized around the pieces of the language. Thus, there are separate chapters on action commands, arithmetic commands, functions, and so on. To help keep you focused on the meaning of a particular word of the *HyperTalk* language, the discussions had to operate in a kind of vacuum. Only with the application examples in the last part of the book is there an attempt to bring the pieces of the languages together.

Dan Winkler believes a good HyperTalk script should look and sound like poetry. Of course, if you set two poets before the same sunset, the poem that each writes will be quite different.

In this part of the *Developer's Guide*, we approach the *HyperTalk* language—including messages, commands, functions, properties, constants, and control (if-then-else and repeat) structures—from a different direction. As a *HyperCard* developer, you may fully understand the inner workings of numerous individual commands but encounter difficulty in drawing together your knowledge of several related aspects of *HyperTalk* in a real application. Therefore, the subjects in this part of the book come from programming questions I've heard since the release of *HyperCard* and from problems I've seen in scripts within stacks from a variety of sources. Even if you're comfortable with *HyperTalk*, a number of the following chapters will offer some insights and suggestions you may not have heard before. At the same time, I don't assume that the following chapters will tackle absolutely every problem you've encountered. These are the predominant ones that I've heard about or inferred from scripts I've seen.

Throughout these *HyperTalk* chapters, I will be stressing compactness of scripts—making as few lines as possible do the most work as possible. Dan Winkler, the person most responsible for the syntax and inner workings of *HyperTalk*, believes a good *HyperTalk* script should look and sound like poetry. Of course, if you set two poets before the same sunset, the poem that each writes will be quite different. Similarly, two *HyperTalk* programmers pursuing the same functionality will likely code the solution differently. In few cases is there "the one best way" to write a *HyperTalk* handler, so it's difficult to pursue perfection in that manner. But if you can refine a handler so that it works faster in fewer lines, then the second generation is much better than the first. It is unfair to you and your stack to slap together a script and ignore it thereafter. Go back to it later, study it, and look for ways to make it simpler, more elegant, more like poetry. The following chapters should help you do that.

A Working Laboratory

To fully understand many of the concepts presented in this part of the book, it is essential that you try out the scripts and simple stacks that will be pre-

Danny Goodman is a freelance writer and has been at the HyperCard frontier ever since March 1986, when John Sculley invited him to follow HyperCard's development from the inside. The result was The Complete HyperCard Handbook. Danny is also the creator of Activision's Focal Point and Business Class.

sented to you. It's a hands-on way of learning that cannot be beat. Simply watching static screens on the pages and trying to imagine what happens when you click a button won't bring the ideas home.

Because you'll be writing a lot of handlers and creating a lot of buttons in the following chapters, I've devised a two-background sample stack that will be used in all hands-on demonstrations. The stack is not a real application. The subjects covered in this part are too diverse to appear in a single application. Rather than force demonstrations into either an information-publishing or -managing stack—or worse yet, try to contrive to build a "real" application that encompasses both types—we'll make copies of the original stack in various chapters to work on numerous *HyperTalk* programming problems and opportunities.

Before we can get started, however, you'll have to build the original stack. The raw material for the stack are in the "Stack Ideas" stack that comes on the *HyperCard Ideas* disk of *HyperCard*. Here's how to make the stack:

1. Open the "Stack Ideas" stack. There is a button on the original "Home Card" that links directly to this stack. Or you may open it via the "Open Stack" choice in the File menu.

2. Click on the right hand pointer until you see card four of the index (Figure 1).

3. Click on the miniature card labeled "Divided Card." This card is named in the stack, so you may also type

```
go to card "divided card"
```

in the Message box.

This card (Figure 2) will be good for demonstrations, because it has five background fields (with which to test various text handling abilities of *HyperTalk*) and only four standard background buttons. The box at the upper right corner holding the fish picture will be a good work area for experimenting with buttons.

4. Choose "New Stack" from the File menu.

5. Type the name "Developer's Guide Master" into the file name field. Be sure the check box "Copy current background," is checked (Figure 3).

6. Click the "OK" button or press the Return key.

The new stack is created, and you are brought to that stack. All card-specific information stored in the prototype card from the Ideas stack disappears, leaving blank text fields and an empty box at the top right. The buttons and their scripts carry over, as does the background script that was in the original.

7. Press the Tilde (~) key or choose "Back" from the Go menu to return to the "Stack Ideas" stack.

8. Type Command-1 or choose "First" from the Go menu to return to the first card of the "Ideas" stack.

9. Locate the miniature card labeled "Address Card 3," and click on that mini-card (Figure 4).

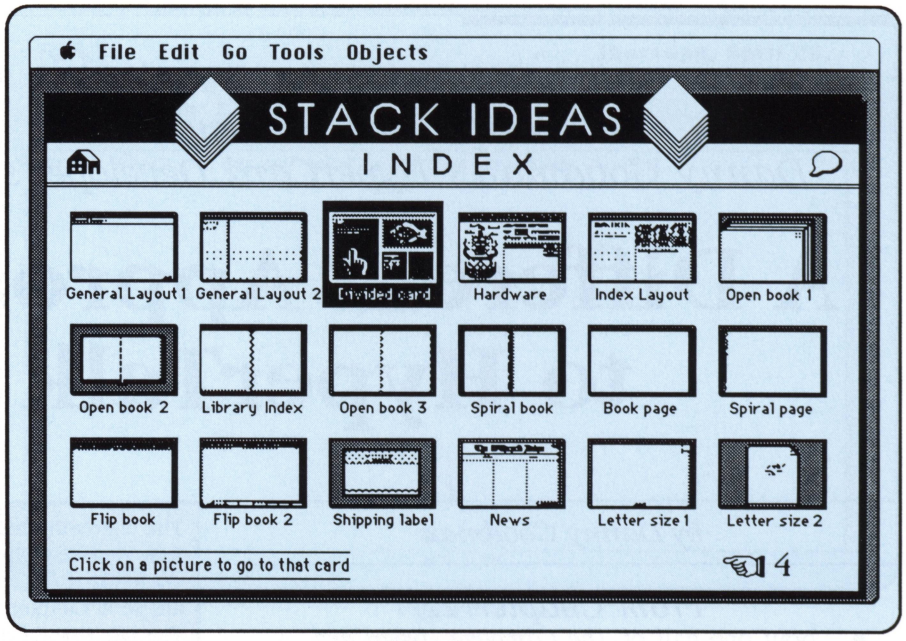


Figure 1
The fourth card of the "Stack Ideas" index.

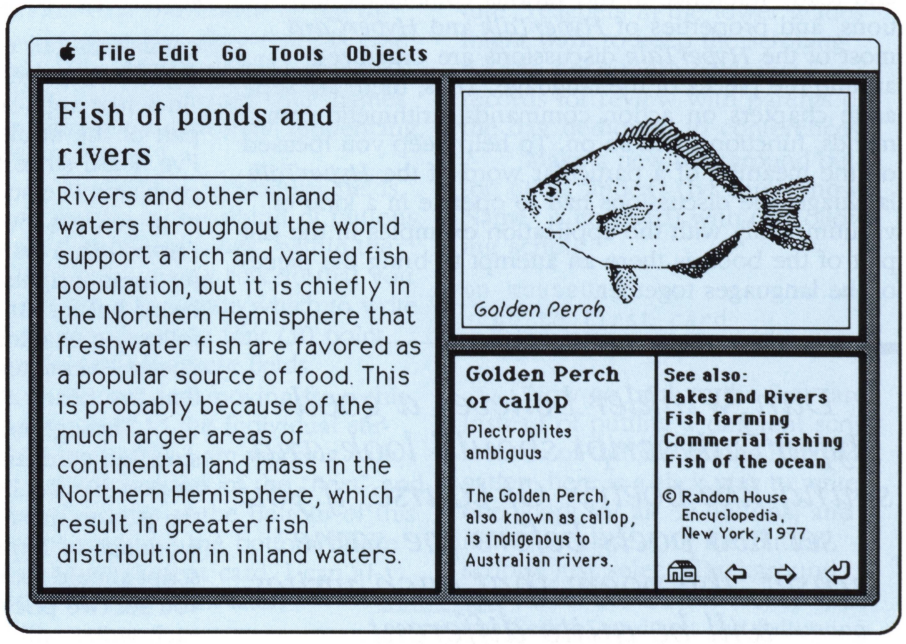


Figure 2
This is the card layout we'll use for all experiments.

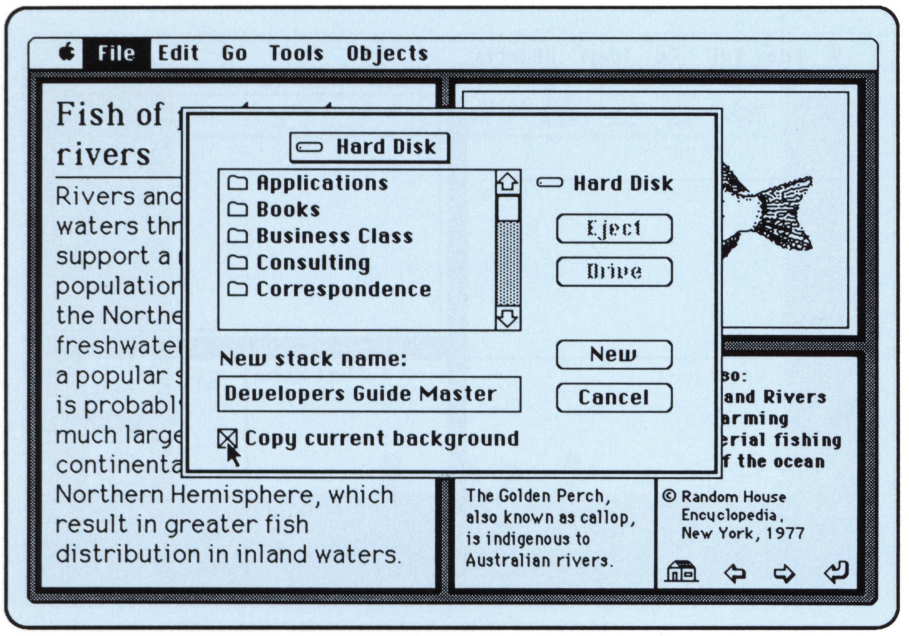


Figure 3

Create a new stack, using this background, and call it "Developers Guide Master."

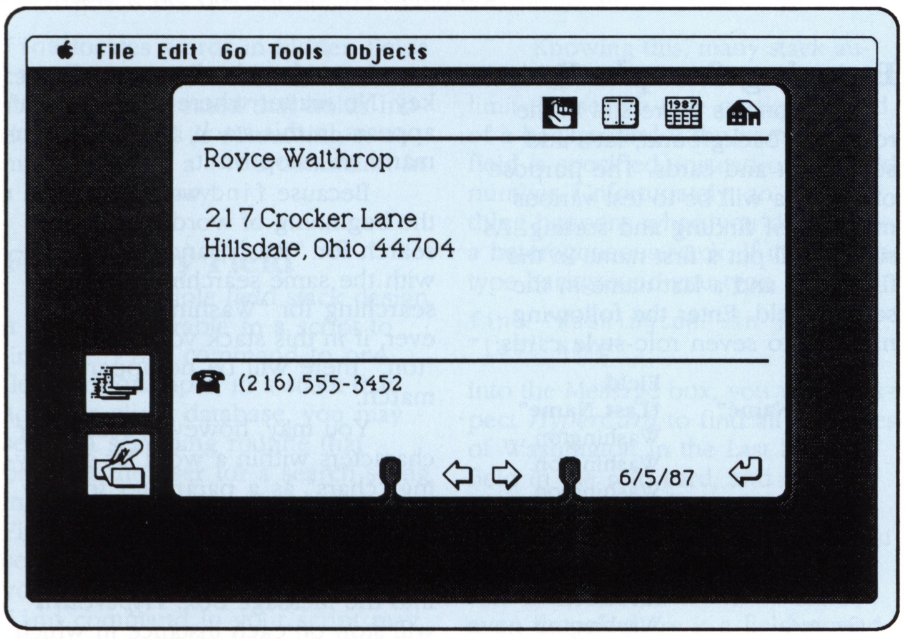


Figure 4

The sample stack will also have a second background—the rolo-style card from the "Stack Ideas" stack.

This address card will become the second background in the new stack you're creating. The card comes with 10 background buttons, 3 background fields, and one hidden card field.

10. Choose "Copy Card" from the Edit menu.

11. Choose "Open Stack" from the File menu and open the "Developer's Guide Master" you just made.

12. Choose "Paste Card" from the Edit menu. This pastes the copy of the address card into the stack. Because we copied the entire card, its card-specific data comes along with it, including the text in fields and the hidden card field. If you want to see that hidden field, type

```
show card field 1
```

into the Message box. When you're finished with the field, type

```
hide card field 1
```

into the Message box.

13. Remove the text from the two large fields.

The script that is part of this card's background automatically inserts today's date into the third field. Leave this script in place.

In most of the chapters of this part, you'll be making a copy of this master stack and modifying it to demonstrate and experiment with various *HyperCard* concepts and techniques. Let's get started with the subject of the *HyperCard* hierarchy, and determining where to put handlers, how to use target names, and how I turned literally hundreds of mouse handlers into just one.

From Chapter 20

Solving Searching and Sorting Mysteries

Among the most perplexing points of *HyperCard* facing stack authors are the fine points of calling `find` and `sort` from within scripts. The problems generally come from expectations that *HyperCard's* searching and sorting should be as complete as dedicated database programs. To the disappointment of many, that turns out not to be the case. But there are some techniques that are not obvious from descriptions of the two commands in *HyperTalk* language manuals. Additionally, stacks with more than one background have also proven troublesome to some authors. In this chapter, we'll attempt to demonstrate the ins and outs of advanced searching and sorting.

Workbook Stack

The best way to understand how `find` and `sort` work is to experiment in a controlled stack environment. For the purposes of this chapter, we'll clone the "Developer's Guide Master" to provide a working laboratory for trying out various commands.

To make the stack:

1. Open the "Developer's Guide Master," created above.

2. Choose "Save a Copy" from the File menu.

3. Type `Chapter 20 Stack` into the File dialog box.

4. Open "Chapter 20 Stack" via the "Open Stack" choice of the File menu.

We'll be using both backgrounds of this stack for demonstration. Several fields need names for the experiments. In the first background, name the first three fields in their Field Info dialog boxes as follows:

```
Last Name
Date
First Name
```

So you know which field is which, choose "Background" from the Edit menu. Then:

1. Choose the text tool from the painting tool palette.

2. As shown in Figure 5, type the names of the fields in the upper-right corners of the fields on the card.

3. Choose the browse tool to take yourself out of background editing mode.

Now go to the second card, which looks like a rolo-style card. Open the Field Info dialog box for the first field and change the name of the field to "First Name." Change the name of the "Phone" field to "Last Name." Then, with text tool from the painting tool palette, type the names of the "First Name" and "Last Name" fields in the upper-right corners of the fields, as shown in Figure 6.

One last setup task before entering sample data is to modify the `closeField` handler in the background script. This handler normally puts `the short date` into the third field of the card whenever one of the other two fields changes. For the purposes of sorting experiments later, the "Date" field needs to receive `seconds` counts instead of `short dates`. Therefore change the handler to read:

```
on closeField
  put the seconds into field¬
  "Date"
end closeField
```

Now we're ready to create a few cards and enter some data to be searched for and sorted.

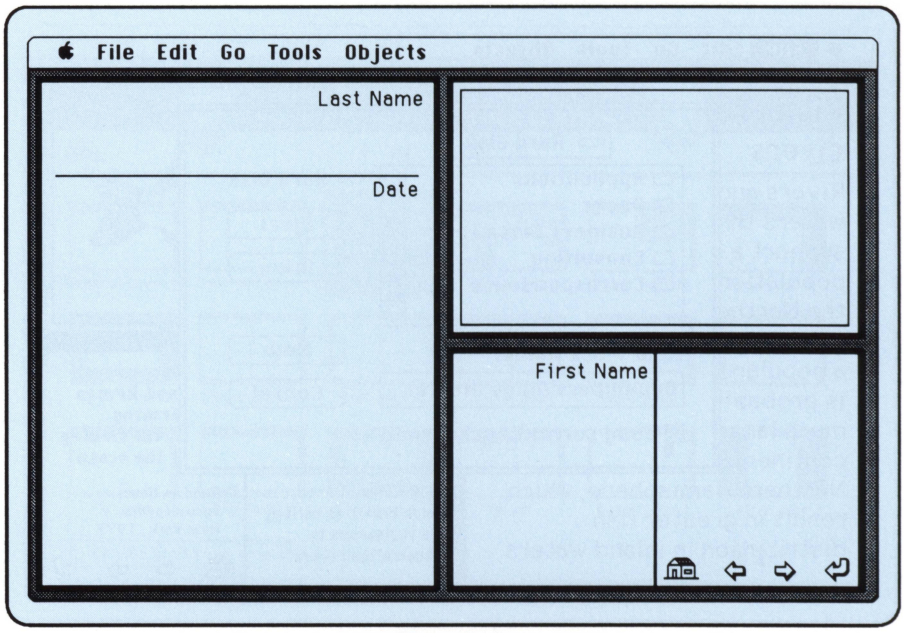


Figure 5

For searching and sorting experiments, name and label (in the graphics layer) the fields of the "Chapter 20 Stack." The field names are intentionally out of order.

Entering Sample Data

As long as we're still in the rolo-style background, let's add some data and cards. The purpose of the data will be to test various methods of finding and sorting. As such, we'll put a first name in the first field, and a last name in the second field. Enter the following names into seven rolo-style cards:

Field "First Name"	Field "Last Name"
Bruce	Washington
Zelda	Washington
Charlie	Washington
Al	Washington
Morrie	Anderson
George	Anderson
Cecille	Anderson
George	Washington

Proceed to the first card of the stack, the single card of the first background. Into the topmost field, "Last Name," type the name "Washington" (without the quotes). We're all set for some tests.

Simple Finding

By now, you surely understand that searching for a chunk of text without any restrictions locates all instances of that text in a stack, even across background boundaries. Type

```
find "Washington"
```

into the Message box, and keep pressing either the Return or Enter key. No matter where "Washington" appears in this stack, the `find` command will stop on it.

Because `find` works only on the beginning of words, you could search for "Wash" and come up with the same searching results as searching for "Washington." However, if in this stack you try to find "ton," there will be no apparent match.

You may, however, search for characters within a word by specifying "chars" as a parameter to the `find` command. Type

```
find chars "ton"
```

into the Message box. *HyperCard* will stop on each instance in which the letters occur in a word. The `find chars` command is much slower than the plain `find` command. If you try this on the "Chapter 20 Stack," you may wonder why the `find` command stops twice on the first card of the rolo-style card, yet you cannot see the rectangle surrounding the letters anywhere. The secret is that there is a hidden card field in the first card, and that field contains some text. To see its contents, type

```
show card field 1
```

into the Message box. The word "button" is the word that stops the `find chars` "ton" operation. That

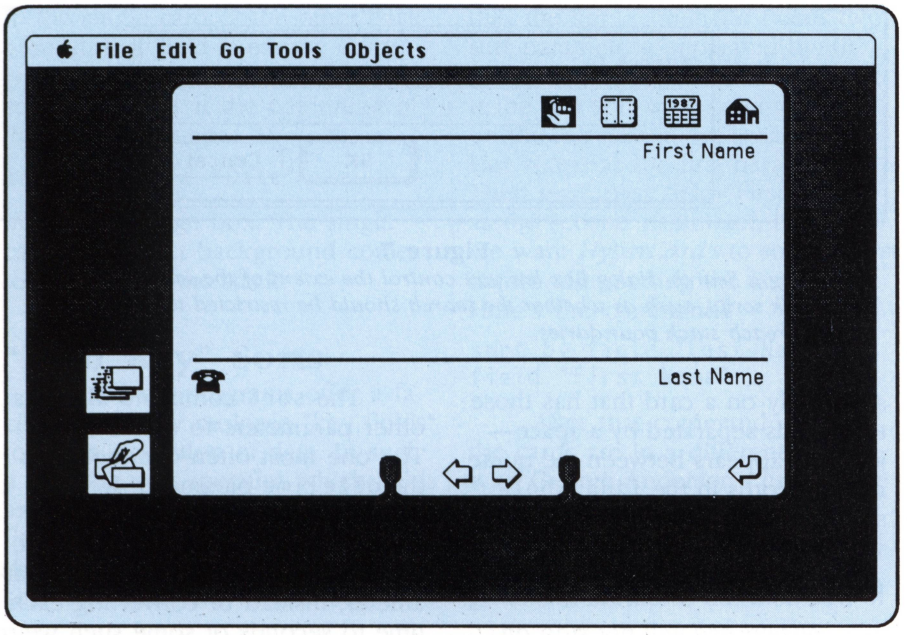


Figure 6

Do the same with the second background in the "Chapter 20 Stack" for only the top two fields.

`find` locates words in hidden fields can sometimes wreak havoc with a stack you've created that holds hidden fields. This is something to remember when a `find` command in a script goes awry.

Finding By Field

In a multiple field stack design, it is often desirable in a script to limit the `find` command to one field. For example, in a stack that acts as a client database, you may set up a searching routine that prompts the user for a search string in an Ask dialog box, as shown in Figure 7. Then the script limits the search to the "Client Name" field. If you don't restrict the search, the `find` command in your script may stop on an identically spelled word in a notes field of someone else's card. *HyperCard*, however, lets you specify a single field to search.

If you search for a text string limited to a particular field number, *HyperCard* will continue to search on that field number throughout the entire stack, even if there are multiple backgrounds, and those backgrounds' fields have an entirely different organization. Thus, the command,

```
find "Washington" in field 2
```

will locate all instances of "Washington" in field 2 of all cards of the stack.

Knowing this, many stack authors figure it should be possible to limit the search to a particular field of a background if the name of the field is specified, instead of the field number. Unfortunately, an unusual thing happens when you do this in a heterogeneous stack. If in the rolo-type background you type

```
find "Washington" in field
"Last Name"
```

into the Message box, you would expect *HyperCard* to find all instances of Washington in the Last Name field in the rolo card, and in the "Last Name" field of the first card of the stack. But when you try it, you discover that *HyperCard* does not find the instance in the first card, even though there is a field named "Last Name." What gives?

Inside *HyperCard*, the `find` command starts its search in the current background on the field whose name you specify in the `find` command. But from there the command is really keying in on a field number. The name becomes irrelevant. Therefore, when the other background comes into view, `find` looks for the string in field 2, not some field called "Last Name."

Here's an experiment that will hammer this concept home. With the card setup specified earlier (make sure "Washington" is entered into the top field of the first card, as shown in Figure 5), type

```
find "Washington" in field
"Last Name"
```

into the Message box while viewing one of the rolo cards. Continue pressing the Return key until you have cycled through the cards at least once, to prove that you'll not stop on the first card with this `find` restriction.

Next, navigate to the first card of the stack (type Command-1 as a shortcut). Duplicate "Washington" in both the "Last Name" and "Date" fields on the card. Now advance to any rolo card and issue the same `find` command from the Message box. Press the Return key slowly to watch what happens.

Eventually you reach the first card of the stack, with *HyperCard* finding Washington in the second field (despite its name being "Date"). If you press Return again, *HyperCard* sends the command anew from this card. Since this card has a field named "Last Name," the command takes over, and finds the instance of "Washington" on this card in field "Last Name." But since this field is number 1 in tabbing order, any subsequent press of the Return key to find "Washington" in field "Last Name" will look for the text in the first field of other cards in this background or others within the stack. Try it. All you get is the first card of the stack.

Find in Field Workaround

This problem plagued one stack design I had. There was no question that the `find` command that a script of mine issued had to limit search to one field in a particular background of a heterogeneous stack. Unfortunately, *HyperCard* does not let you limit a search to a single background. It's the whole stack or none.

I felt it was like cheating, but I ended up hiding a field in the other backgrounds so that the `find` command would not stop on any cards in those backgrounds. I had to make sure that the empty, hidden fields in other backgrounds had the same field number (in tabbing order) as the field that was being searched for in the primary background. Because I discovered this late in the

development cycle, the new field in each of the secondary backgrounds had to be pushed farther into position. But once in place, the `find` command from the primary background worked like a charm.

Incidentally, there seems to be a bug in all versions through 1.2 that makes `find` unusable if you're trying to limit a search to card fields. The `find` will stop on matches in background fields as well. Therefore, until this is fixed, avoid searches that must be restricted to card fields.

Boolean Finds

The plain `find` command in *HyperTalk*, at least through version 1.2, allows only what is called "and" searching, but not "or" searching. "And" searching means that if you specify two strings within the quotation marks as parameters to the plain `find` command, *HyperCard* will stop only on those cards that contain *both* strings somewhere in the fields of the card.

Some authors have asked about doing "or" searches. An "or" search would let you specify two strings, but *HyperCard* would find the incidence of either string on a card. So far, *HyperCard* does not allow that, nor have I seen any handlers that might replicate an OR search. Perhaps it's possible. I'd love to see it.

Find Whole and Find String

HyperCard got a boost in its `find` capabilities with the release of version 1.2. Perhaps the most important addition is the `find whole` command (also activated from the keyboard by pressing Command-Shift-F). Whereas the plain `find` command is sensitive only to word starts, the `find whole` command is sensitive to word starts and endings. Therefore, you may now search for the occurrence of two or more complete words when they appear together. For instance, if you wanted to find the state name "New Mexico," the simple `find` command would stop on any card that had the words "new" and "Mexico" scattered anywhere on the card. But

```
find whole "New Mexico"
```

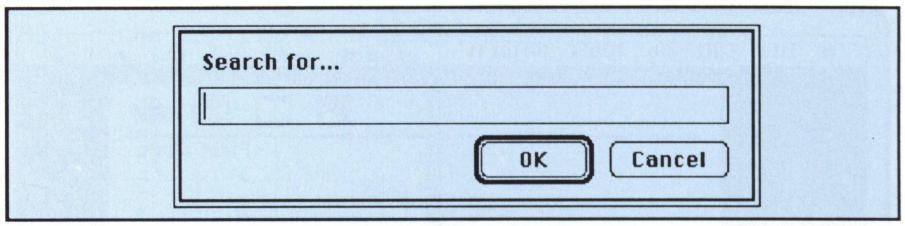


Figure 7

Presenting a Search dialog box lets you control the extent of the search in a HyperTalk script, such as whether the search should be restricted to one field or should breach stack boundaries.

stops only on a card that has those two words separated by a space—just as it appears between the quote marks. Words in the `find whole` string must be the complete words you're looking for. If you try

```
find whole "New Mex"
```

HyperCard will not stop on "New Mexico," because the second word does not end "Mex." The good news, too, is that `find whole` uses the same fast searching technique of the plain `find` command.

The second addition with 1.2, `find string` , lets you search for any sequence of characters that span multiple words. Not sensitive to word starts or endings, the command

```
find string "w M"
```

would stop on "New Mexico" as well as on "how many." You would still use `find chars` when the characters are in one word.

To get the best performance out of the `find string` command, specify at least three characters after the space. This triggers *HyperCard's* fast searching technique.

While *HyperCard's* `find` has a number of quirks to it, its `sort` is more straightforward.

Plain Sorting

HyperTalk's `sort` command demands a parameter telling it what field or chunk within a field to sort on. For instance, in a rolo-style stack, you can enter people's names in the normal first and last name order, as they might appear on an envelope. Then you can sort the stack by the last word of the first line of the "Names" field. The `sort` command is intelligent enough to use only the last words of the first line of that field to sort.

The `sort` command also has other parameters to aid sorting. The one most often overlooked is the `dateTime` parameter. If you wish to sort cards according to dates or times in a particular field, you may specify `dateTime` as a parameter, instead of converting each time to seconds or some such weird thing. A command with this parameter would look like

```
sort dateTime by field "Date"
```

The *Handbook* spells out the workings of these parameters in detail.

Sorting by Field

Unlike searching, *HyperCard's* sorting works the way you'd expect when sorting by named fields. Let's try it with "Chapter 20 Stack."

If you followed directions to set up the stack earlier in the chapter, you have a few cards with the name "Washington" in the Last Name field, and some with "Anderson." You also have one other card in a different background, with the name "Washington" also in a field named "Last Name." This "Last Name," field, however, is field 1 in tabbing order, while all other cards place the "Last Name" field as 2 in the tabbing order. Type

```
sort by field "Last Name"
```

into the Message box and press Return. An amazing thing happens. The lone card of the other background becomes mixed with the other cards based on the Washington value in the "Last Name" field. One of the "Anderson" cards is now the first card in the stack.

You may restore the stack to its original order by sorting on the "Date" field. In the rolo cards, this field contains the seconds values for the last updates to those cards.

In the single card of the other background, that field is empty. Cards with empty fields in a field-specific sort are placed at the beginning of the stack. Try it now. Type

```
sort by field "Date"
```

into the Message box. The single card of the first background comes to the front of the stack.

"Dual Key" Sorts

Database programs offer a facility to specify multiple "key fields" to sort a collection of data. In such a sort, the user specifies which fields are the primary and secondary key fields. For example, if you have a database with first and last names in separate fields and wish to sort the collection, you would specify the last name as a primary sort field, and the first name as the secondary sort field. That way, the database sorts initially by the last name. Then within records containing the same last name, the first names would be set in alphabetical order.

Unfortunately, HyperCard does not let you limit a search to a single background. It's the whole stack or none.

You can specify multiple value sorts in *HyperTalk* without much difficulty. While the slow way would be to sort the stack twice—once on the primary sort criteria, once again on the secondary—the `sort` command allows multiple parameters.

To experiment with multiple sorts, let's be sure the "Chapter 20 Stack" is alphabetically mixed up. Type

```
sort by field "Date"
```

into the Message box. This will put the stack in order based on the original entry order (unless you modified one or more cards since then), since the sort is on the seconds in the "Date" field.

Our goal with the next `sort` command is to sort the rolo cards alphabetically so that all the Andersons are in front of the Washingtons, and each of the groups is alphabetized by first names. To do this, the `sort` command needs two sort parameters, separated by an ampersand. Since according to logic the first sort we'd need would be on the last names, we'll pass field Last Name as the first parameter. Then we'll pass field "First Name" as the second parameter, because we want *HyperCard's* to sort the first names after sorting the last names. Here's the command:

```
sort by field "Last Name" &
field "First Name"
```

After that command executes, the cards are in a different order, with George Anderson's card at the front of the stack. As you flip through the stack, you'll notice that the lone card of the other background is sorted as the first card of the Washington group. That's because, you'll recall, the field First Name is empty on this card, and thus sorts first in the group.

You may also sort on multiple keys in reverse sort order by inserting the `descending` parameter. To flip the order of the cards just sorted, type

```
sort descending by field
"Last Name" & field "First
Name"
```

into the Message box. Zelda Washington's card is now the first card of the stack.

Sorting Card Suites

As we discussed in Chapter 18, you may organize a heterogeneous stack such that a very specific set of cards are linked together in a linked card suite. One of the supreme advantages of linking cards in this manner is that a sort does not harm the user's navigation to the cards. Links are hard-wired into the system, so sorting order is completely transparent to the user.

However, the author may wish to sort the cards for other purposes. For instance, you may want to collect an alphabetized list of clients from a heterogeneous client record stack for use elsewhere in your stack system. Such a sort would also keep the summary cards in alphabetical order for the user who wishes to browse through the summary cards for a client.

If you sort a card suite by a field name of the summary card, there is the danger that the first card of the stack after the sort will not be of the summary card background. This would happen if the other backgrounds do not have a field with the name of the sort key, or they have such a field, but the fields are empty.

When the other backgrounds have the same field name, it is incumbent upon the author to maintain control over the stack so that those fields are never empty prior to a sort, while the named field of the summary card has data in it. It probably requires posting of data from the summary card to the other backgrounds, but it must be done by the script to assure consistency and proper sorting.

Once a heterogeneous stack is sorted by field name, the cards could appear to be substantially jumbled if you were to do a sequential navigation through the stack. As long as your linked suite links are in good shape (and sorting doesn't touch them at all), then the linear order of the stack should be of no consequence. But if you need to retrieve data from a particular field in an alphabetical sequence of cards, use a repeat loop to go to the next card of the same background until you've made the entire circuit.

I hope that the above explanations and examples have cleared up the mystique surrounding *HyperCard's* `find` and `sort` commands. With any luck, I've also given you some organizational ideas for your next stack.

Create a Smart Field for a HyperCard Stack and Win $50

Fields of Intelligence

If you thought fields were for raising horses, playing baseball, or just storing text, then throw away all of those old-fashioned ideas because here come the "HyperFields." *HyperCard* fields are far more powerful than what normally comes to mind when you think of a field in a database program. In fact, *HyperCard* fields can do almost everything buttons can—plus they do a lot of things buttons can't. This column is devoted to showing how fields can flex their muscles, and thus help improve your stacks.

First set up a field with a list of commands, placing each command on a separate line (see Figure 1). Be careful that none of the commands wraps to the next line. When you click on the command inside the field you will immediately execute that command. The command can be any legal *HyperTalk* command, or any user-created command in the *HyperCard* hierarchy. This includes commands called by routines that you have written. For example, you may create a command called `look` and place it in the background script for a stack, then call the routine named `look` when it is clicked on in our intelligent field.

Here's what you do to create this field. First create two as illustrated in Figure 1. Both of these fields must be background fields, or the scripts will not work properly. Name the larger field "Commands" and enter the following into its script:

```
on mouseUp
  put item 2 of the clickloc into vClick
  put item 2 of the rect of field "Commands"¬
  into vLoc
  put the scroll of field "Commands" into ¬
  fScroll
  put the textheight of field "Commands" ¬
  into fHeight
  put trunc((vClick - vLoc + fScroll) ¬
  / fHeight ) + 1 into selected
 put selected into message
 get line selected of field "Commands"
 do it
end mouseUp
```

This routine could be shortened, but using descriptive variable names makes the function of the script as clear as possible. The parameters like `textheight` could be hard coded, but that would limit the routine's flexibility.

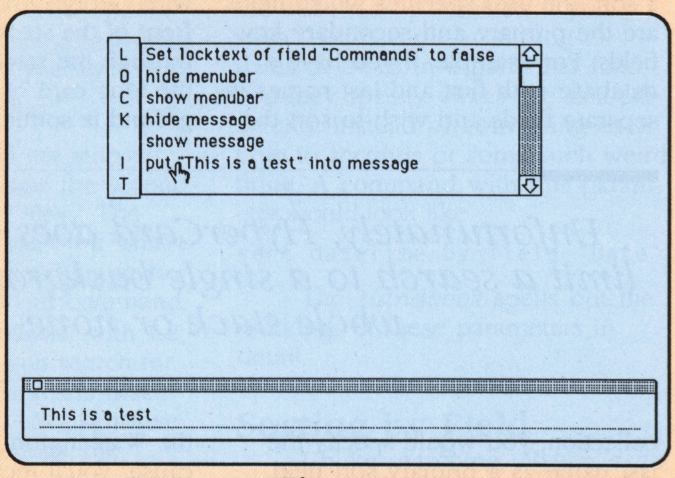


Figure 1
This field executes any command in the field on which you click.

The second, smaller field, has a simple script:

```
on mouseUp
  set locktext of field "Commands" to true
end mouseUp
```

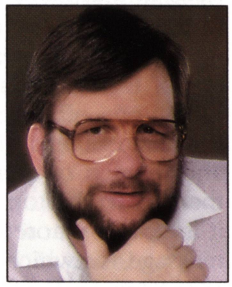
It is this field's responsibility to lock the text in the "Commands" field so the latter behaves like a button. The `locktext` function of the "Lock It" field should always be set to true. Notice that this smaller field could have been handled with a button. The button, however, couldn't have the words "Lock It" spelled vertically as this field does. Different text formats is just one plus of using fields for buttons.

Be aware that because these fields are background fields, creating a new card in this background duplicates the fields and their scripts. It does not, however, duplicate the text in the fields, so you need to reenter the commands on any new cards.

Send in Your Smart Field

Send us your own intelligent field. If we select it for publication, you'll win $50 and get your name in the pages of *HyperLink* as the author. Send a disk with the stack containing your field and text explaining how it works to: *HyperLink Fields of Intelligence*, P.O. Box 7723, Eugene, OR 97401.

Shafer On Scripting



Featuring Radio Button Management Techniques

by Dan Shafer

One of the *HyperTalk's* really powerful ideas is the concept of "abstracting" scripts. Putting handlers as high up the *HyperCard* hierarchy as possible, generalizing as you go, will make your stacks more efficient and make you a far more productive scripter. In this column, I'll take a look at this idea and how it can affect scripting strategies. I'll also discuss how to group and handle radio buttons efficiently, which is one of the topics I promised to cover in my premier column.

Of Hierarchies and Messages...

As I have frequently pointed out here and elsewhere, *HyperTalk* is an object-*like* language. It isn't completely object oriented in the way Smalltalk is. But because it adopts some of the same key ideas Smalltalk does, many of the programming strategies that apply to object-oriented languages can be put to good use in *HyperTalk*.

HyperCard, as you probably know, has a definite hierarchy through which messages pass. This hierarchy is depicted in Figure 1. It starts with buttons and fields and works its way upward to *HyperCard* itself. Most of the messages your scripts deal with are system messages generated by the user interacting with your stack -- pushing buttons, for the most part. As you know, each button-press generates a *HyperTalk* message—`mouseUp`, for example.

Each message is sent to a specific initial target object. for instance, the `mouseUp` message goes, logically enough, to the object (usually, but not always, a button) in which the event occurs. The `openCard` message is sent to the card being opened. If the object receiving the message has a handler of the same name as the message, then it handles the message itself. If it does not have an appropriate handler, it passes the message up the hierarchy to the next level. There, the next object in the hierarchy looks to see if it has a handler to deal with the message. If so, it carries out the instruc-

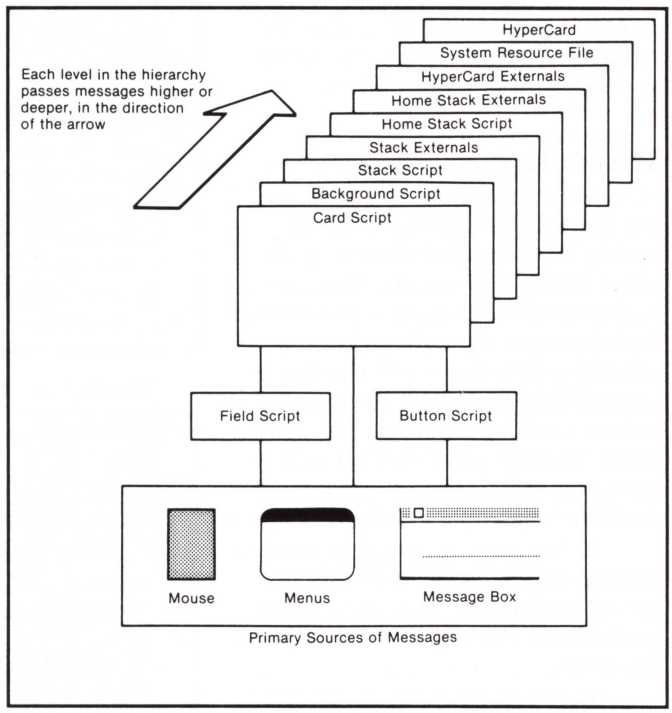


Figure 1
The HyperCard Message Hierarchy

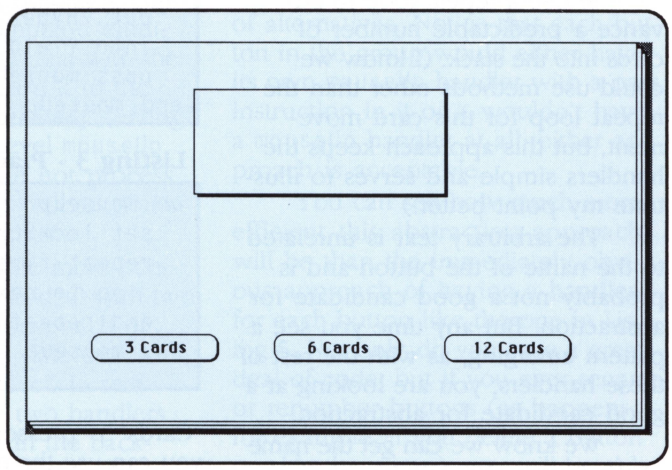


Figure 2
We'll use this card as an example of abstracting scripts—a scripting technique that maximizes efficiency in stack development.

Dan Shafer is the author of many microcomputer books, including HyperTalk Programming, published by Hayden Books Division of Howard W. Sams & Co.

tions in that handler; if not, it passes the message up the hierarchy.

When the message gets to the top of the hierarchy—to *HyperCard* itself—one of three things happens. If the message is a system message and there is no handler for it along the way to the top of the hierarchy, it is simply ignored. If it is a system message for which a handler exists at the top level of the hierarchy, that handler executes. If the message is *not* a system message, but one you have defined, an error results because *HyperCard* has no idea what to do with it.

With that background in mind, let's examine how we can take advantage of *HyperCard's* hierarchy to make scripts more efficient.

Move It Up!

Here's a simple example for the purpose of illustration. Let's build a card like that shown in Figure 2, with three card buttons and a background field named "Content." Each button's job is to put some specific information into the background field and then advance a certain number of cards into the stack. The first word in each button's name defines the number of cards to move into the stack.

A first pass at scripting these buttons might result in the scripts shown in Listing 1. As you can see, each handler consists of three basic functions:

1. Put some arbitrary information into field "Content."

2. Lock the screen.

3. Use a repeat loop to advance a predictable number of cards into the stack. (I know we could use methods other than the repeat loop for this card movement, but this approach keeps the handlers simple and serves to illustrate my point better.)

The arbitrary text is unrelated to the name of the button and is probably not a good candidate for abstraction. But any time you see a pattern emerging, as with the rest of these handlers, you are looking at a good candidate for abstraction.

We know we can get the name of the button that receives the message and use it. That's the purpose of the *HyperTalk* function `the target`. When the button called "Cards" is pressed, for example, you can use the expression `the name of the target` to put the object's full name somewhere. You can also use `the short name of the target` to eliminate some housekeeping data from the button name. (In this case, for example, `the name of the target` returns the string "card button 3 Cards" while

Listing 1 - Handlers to be placed in Card Buttons 1, 2, and 3 respectively

```
on mouseUp
  put "This is an interesting message" into field ¬
"Content"
  set lockScreen to true
  repeat 3 times
    go to next card
  end repeat
end mouseUp

on mouseUp
  put "You can do a lot with HyperTalk!" into field ¬
"Content"
  set lockScreen to true
  repeat 6 times
    go to next card
  end repeat
end mouseUp

on mouseUp
  put "HyperCard's hierarchy is quite straightforward."¬
into field "Content"
  set lockScreen to true
  repeat 12 times
    go to next card
  end repeat
end mouseUp
```

Listing 2 - Modified handlers that replace those in Listing 1

```
on mouseUp
  put "This is an interesting message" into field ¬
"Content"
  pass mouseUp
end mouseUp

on mouseUp
  put "You can do a lot with HyperTalk!" into field ¬
  "Content"
  pass mouseUp
end mouseUp

on mouseUp
  put "HyperCard's hierarchy is quite straight-forward."¬
  into field "Content"
  pass mouseUp
end mouseUp
```

Listing 3 - Place in Background script in conjuction with Listing 2

```
on mouseUp
  set lockScreen to true
  repeat first word of the short name of the target
    go to next card
  end repeat
end mouseUp
```

Listing 4 - Place in either Background, Card, or Stack Script for more efficient radio button management

```
on mouseUp
  repeat with counter = 1 to 4
    if counter = the number of the target then
      set the hilite of button counter to true
    else
      set the hilite of button counter to false
    end if
  end repeat
end mouseUp
```

Listing 5 - Placing scripts like this one in each button is much less efficient than using a single script as in Listing 4

```
on mouseUp
  set the hilite of me to true
  set the hilite of button 13 to false
  set the hilite of button 16 to false
  set the hilite of button "Test" to false
end mouseUp
```

`the short name of the target` returns the string "3 Cards".) Let's take advantage of that fact and see if we can generalize a handler that will do what we want.

A quick look at the handlers reveals that the number that appears as part of the button's name is used as the counter in the repeat loop. (This is quite often true, if you name buttons judiciously.) So we generalize the repeat loop like this:

```
repeat first word of the ¬
short name of the target
  go to next card
end repeat
```

We can also, obviously, move the set lockScreen to true line into this more generalized handler.

Having identified what pieces of the handler lend themselves to abstraction, we can now create the basic handler each button will have. Listing 2 shows the reduced handler for each of the three buttons. Notice that each handler has some processing to do that is independent of its name or position (the `put` statement) and then uses the `pass` command to force the message up the hierarchy. The `pass` is necessary because a handler that deals with a message doesn't pass it up the hierarchy unless specifically instructed to do so. Usually we only want one handler to deal with any message.

Now we simply need to install a handler at least one level up the hierarchy that will handle the common processing necessitated by these `mouseUp` handlers as they pass the message. This handler must have the same name as the message that is being passed, of course (in this case, `mouseUp`). So let's put the handler in Listing 3 into the background of the stack. We could choose to put it into the stack itself; the choice is generally somewhat arbitrary. (Note that we could not put this handler at the card level, even though that is higher in the hierarchy, because doing so would make the handler inaccessible to the background buttons on any other card in the stack. The buttons would not function as expected.)

Sometimes when I describe this process, a question arises. "What if I have one or more other buttons that I don't want handled this way?" The answer is simple: just don't use the `pass mouseUp` command. With that command eliminated, the button will deal with the message and not return it to the message stream, effectively "eating" it. The background-level `mouseUp` handler will therefore not process the message; it can't process something that never reaches it.

Things get a little more complicated if you want to deal with two or more sets of buttons, each of which needs slightly different treatment at the higher level. In that case, you can't have two handlers with the same name in the background, so use an `if` test to determine which handler to use.

From this simple example, you should be able to go through your scripts and look for opportunities to generalize. You can see how much more efficient such approaches can make your scripts. In this case, we started with three handlers having a total of 21 lines and reduced them to four handlers with a total of 18 lines. That isn't a major saving in absolute terms but it represents almost 15% less code. In a real-world situation you can see that a 15% reduction in substantial portions of the program is quite significant.

Managing Buttons

Advanced scripters frequently want to deal with groups of radio buttons and get stumped by the various ways one could approach the problem. Using the abstraction principles covered in the first part of this column and a little clever logic, you can create a radio button handler that is extremely efficient.

First, make sure that the radio buttons you want to work with are all in a consecutively numbered group in the background or card. You can manage radio buttons that don't exist as a nice contiguous group but it isn't as straightforward. You can use the *HyperCard* "Send Farther" and "Bring Closer" commands from the Objects menu to rearrange buttons to achieve this grouping.

Once you have grouped the buttons appropriately, you need only to write a `mouseUp` handler at the card, background or stack level (depending on design considerations) to turn the appropriate buttons on and off. Listing 4 shows the method I use for this; there are lots of alternatives. Notice that each button in the group would either have its own `mouseUp` handler with a pass instruction in it or it wouldn't have a `mouseUp` handler at all; either approach is acceptable.

You can see how much more efficient this abstraction approach will be than the immediately obvious approach of having a handler for each button like the one in Listing 5. Not only do you save a great deal of code, but if you ever rename or renumber buttons (as happens, for example, if you delete a button outside the group), you will be able to revise the script much more easily if you use this approach.

Send HyperLink Your Best Button, Win $50, and Take a Bow!

Buttons & Bows

This issue's winning button is one of those deceptively simple, very useful tools. After you read this, you might think, "I could have written that in less than a minute." But Mark W. Richmond of Austin, Texas *did* write it and submit it to us, so he wins.

There are many times when creating stacks that you need to find a particular screen location. This location is expressed as a point: two numbers separated by a comma. The first number is the horizontal location; the second number, the vertical. For example, the bottom-right corner of the Mac Plus screen is 512,342.

The exact location of a given point on the screen can be a very important piece of information when scripting. For example, if you wish to do a `drag` command to do some painting from within a script, you pass the starting and ending points of the `drag` as a parameter. Here's an excerpt from a script that uses the brush tool to draw a line:

```
choose brush tool
drag from 20,25 to 35,45
choose browse tool
```

As long as you know the proper screen co-ordinates, the above script is simple. But pinpointing the exact location for the `drag` command can be, well, a drag. But with this button and its accompanying field in your library of *HyperCard* tools, it's a snap. Just copy them on to the card you're working on, and the screen co-ordinates are just a click away.

How To Use It

To use this card button all you do is click on it once and the current mouse location is placed into the card field called "where." As you move the mouse, this value changes continuously to effect the change in position. In Figure 1, this field is located directly to the left of the button. To stop the action, just click the mouse button; the value displayed in the "where" field corresponds to the position of the mouse. All other action on the screen is suspended while this script is running, and the only way to get control again is to click the mouse.

If you want to put the location directly into a script you can select it, "Copy" it to the clipboard, open the script where you want to place the value, and "Paste" it where you want it.

Listing 1

```
on mouseUp
  set cursor to 2
  repeat until the mouseClick
    get the mouseLoc
    put it into card field "where"
  end repeat
end mouseUp
```

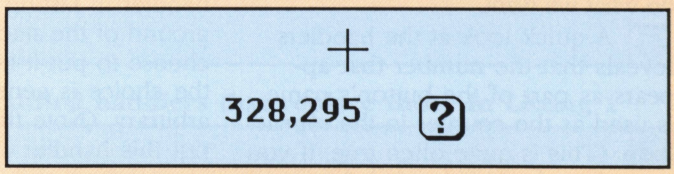


Figure 1

How It Works

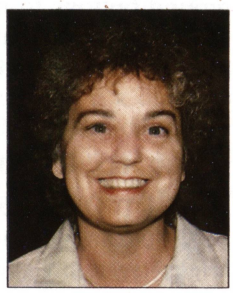
Here's all you have to do. First create a small button and assign the question mark icon to it using the "Icon" button in the "Button info" dialog box. Then create a new field large enough to hold six numbers and a comma and name it "where" in the "Field info" dialog box. Then enter the script in Listing 1 into the script of the button.

Here's what the script does. After setting the cursor to the cross hair (2), a `repeat until` loop is entered that continues until the next `mouseClick`. Two things happen in this loop. First, we `get` the current `mouseLoc`, then we `put it into card field "where"` which displays it.

Send in Your Button

Do you have a button idea to share with *HyperLink* readers? Send a copy on disk with a letter explaining what it does to: *HyperLink Buttons & Bows*, P.O. Box 7723, Eugene, OR 97401. If we select it for publication, not only will your name become nationally known, but you will also receive a check for $50 from *HyperLink*. One piece of advice: We're looking for *HyperTalk* scripts that can be keyed-in from the magazine. Please use *HyperCard's* built-in commands and functions, or those you create within *HyperCard*. Scripts that rely on XCMDs and XFCNs are not appropriate for *Buttons & Bows*.

In The Cards



Using Multiple Backgrounds...

by Carol Kaehler

In the first issue we talked about the fact that a stack with just a single background can still produce a rich fabric. Even with a single background, there's the division between card objects and background objects. Fields, buttons, and pictures can appear on individual cards or in a background that lets them be shared among many cards. And within a card or a background, objects can be either transparent or opaque, showing through to or obscuring any object underneath. But even with all of this flexibility, a single background sometimes isn't enough for a stack. I first noticed this during the development of the "Help" stack, and that is why Bill Atkinson put multiple backgrounds into HyperCard.

When Bill created multiple backgrounds, I was able to have a single picture that served as a background for all cards in the section.

When One Background Isn't Enough

The "Help" system has 19 backgrounds in the main stack. Why are there so many different backgrounds in the "Help" stack? You create a new background whenever you want a group of cards within the same stack to have its own set of common characteristics.

If there are individual cards that require a different background picture, you can create a card picture on the individual cards that require it. But the design of the "Help" system dictated that every card within a section have the same "tab" showing frontmost in the notebook. Because each section might have 50 or more cards in it, this meant that I had to copy the picture over and over to each of the cards in the section. If the tab had to be altered, I had to alter each of the cards individually. And if a particular card had additional art on it, I had to carefully preserve those differences as well. When Bill created multiple backgrounds, I was able to again have a single picture that served as a background for all cards in the section.

Each main section of "Help" uses a different background picture displaying the corresponding subject--such as Browsing. Figure 1 shows the background picture for the Browsing section of "Help." By the way, you can see any card's background by choosing "Background" from the Edit menu.

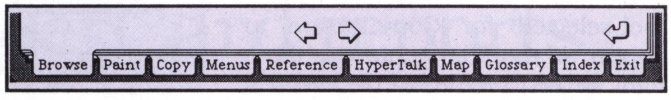


Figure 1

Modifying the tabs on the "Help" notebook from background to backgraound makes it easy to identify the different sections of a stack.

Multiple Backgrounds for Multiple Buttons and Fields

The various backgrounds in the "Help" stack also contain buttons that do different things depending on which section of "Help" you're in, and they have fields that lay out the text in a way that's appropriate for that section. For example, the "Index" card uses two columns of text side-by-side instead of the one column used by most of the "Help" system.

There are several backgrounds within the Menus section. I did this so I could have a separate background for each menu. I wanted each menu to be shared only among the cards that described it. By creating separate backgrounds for each menu, I could not only share the menu picture, but create background buttons over each one of the menu commands to take you to the card that described that command. If a menu had 12 commands, I could create just 12 back-

Carol Kaehler is the author of the Mac Owner's Guide, MacPaint manual, and the recently released book HyperCard Power: Techniques and Scripts. As a key member of Apple's HyperCard development team, she wrote the HyperCard on-line "Help" system.

ground buttons to move you among the cards rather than 144 card buttons. This is a real time and space saver.

How to Create More Backgrounds?

Start in a section of the "Help" system's spiral notebook whose section design you'd like to copy. (Or use any existing stack you like by going to it.) This will be the basis for all of the sections in your new stack. Choose "New Stack" from the File menu and name your new stack. Leave the "Copy Current Background" button checked so your new stack will start out looking like the background the stack you're copying.

When you're ready to add a new background, choose "New Background" from the Objects menu. You get a blank background picture. Then to paste the original background picture into the new background., first use the left-arrow key to return to the card with the original background. Next, copy its picture by either using the selection tool (you can select an entire picture by double-clicking on the selection tool) or choosing "Copy Picture" from the Edit menu. Remember you must have a Paint tool selected for "Copy Picture" to appear on the Edit menu. Use the right arrow to return to the card that has your new background, and choose "Paste" from the Edit menu.

Modifying the New Background Picture

Modify the new background picture using the Paint tools. (Make sure you choose "Background" from the Edit menu first.) It's very important not to change the part of the picture that you don't want modified. For example, don't alter the spiral or the edges of the notebook. This ensures a smooth transition from one background to the next. And don't move one of the background pictures accidentally. If they're misaligned even a little, the picture will jump as it passes by.

If you're using the "Help" system as a model, modify the tab to show the current section frontmost on the notebook, and edit any tab currently frontmost so it looks like it's

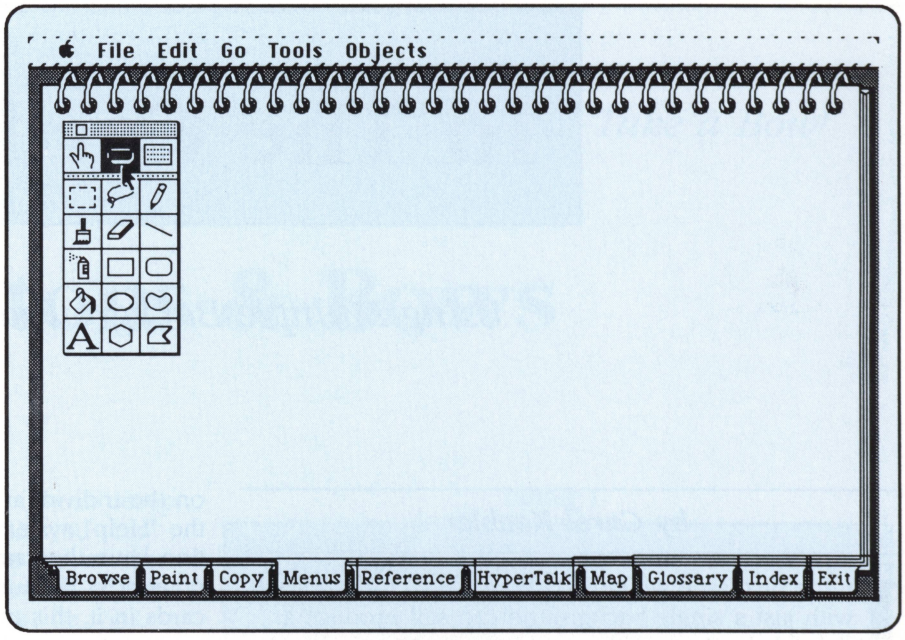


Figure 2

The Background from the Menus Section showing the 10 Background buttons.

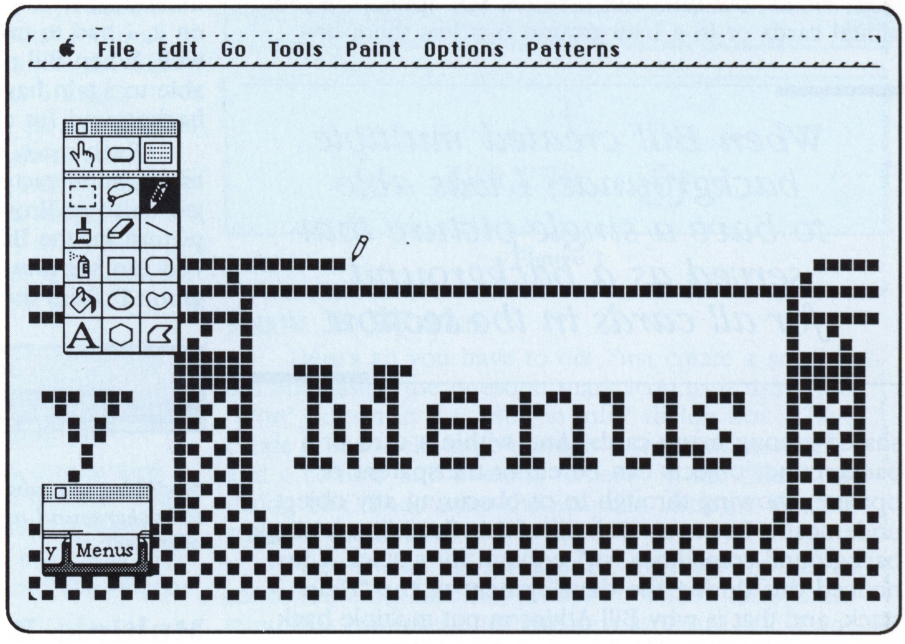


Figure 3

Here's the way the tabs are modified in FatBits to customize a background.

back further in the notebook. You'll probably need to use FatBits to do this. Make the tab's shape the same as others. Don't modify any tabs except those you're moving . Again, this gives consistency to the notebook as you move from a particular card to another in a different section.

Other Reasons to Use Multiple Backgrounds

Multiple backgrounds give you lots of freedom in using different background pictures within a single stack. But using multiple backgrounds also gives you more freedom in using buttons and fields. Let's take a look at another example, again in the "Help" system.

Most of the "Help" system uses one text field for the main text and one for titles. For most of the system this was what I wanted. But in the Glossary and the Index I wanted to use two columns of text on each card. And I wanted to use a different font than I used elsewhere. Because

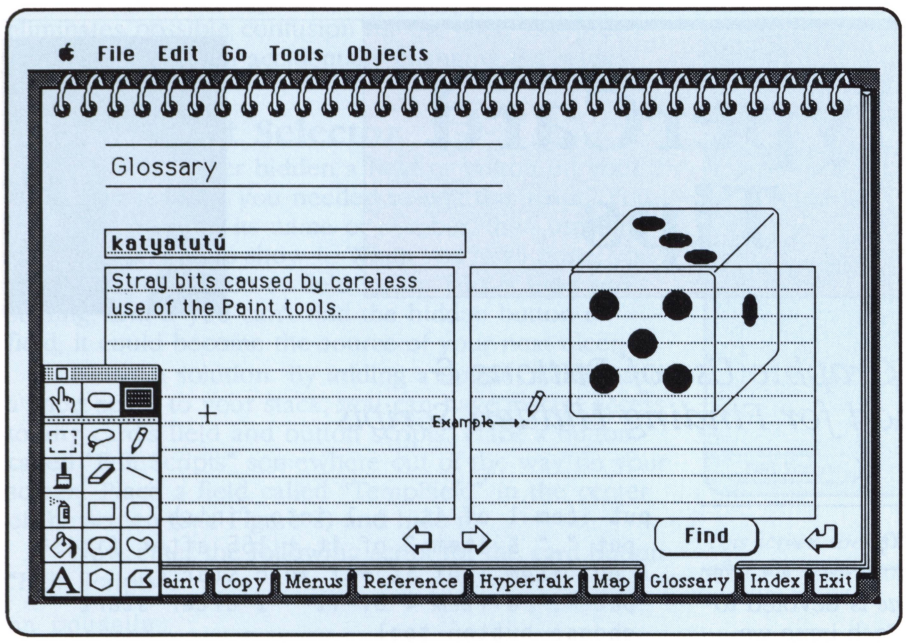


Figure 4

Here the Field Tool is selected in one of the cards in the Glossary section of the "Help" stack. Notice that using transparent background fields allows the integration of art for each card.

I had put each section (each tab) in a separate background as described above, I could put fields and buttons in the background of each separate section, thus getting just enough generality without affecting every card in the stack.

If the whole "Help" stack were just one background, I could have just added new fields to the ones that were already there in the original background. That, however, would have made it confusing when deciding where to click, and the I-beam might show up in unexpected. Instead, in the new background I deleted the original field and in its place I created two side-by-side fields, selecting new fonts for them (with the field selected with the field tool, you can double-click to see the field's dialog box.)

More Than One Set of Background Buttons

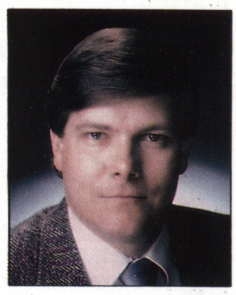
Sometimes you want buttons to do different things, depending on what part of the stack you're in. Either you want a different look for the button, or you want it to do something different than the original. You can clone the existing background onto the new background by copying and pasting it. Next, double-click the new button to see its Button Info box. Click "Script" and replace the existing script with the new one you want it to have. Now the button will do different things, depending on which background you're in.

You can also move the button to a different place or give it a different look for just this background by clicking various options in its Button Info dialog box.

You can see how many backgrounds are in your stack by choosing "Stack Info" from the Objects menu. And you can find out about the current background by choosing Bkgnd Info from the Objects menu. To go quickly to the next background, type "Go next background" into the Message box and press the Return key. To move to the next card that shares a background, type "Go next card of this bkgnd" into the Message box and press Return. You could also type "Go first card of next (or prev) background" into the Message box.

To delete a background, delete all the cards that use it. By creating different backgrounds that have similar elements, we combine the unity of a similar look and feel throughout a stack, with the added functionality of tailoring each background to the tasks at hand.

HyperCard Tips



A Graphic Use of Buttons & A Method for Finding Hidden Scripts

by William K. Balthrop

Every issue of *HyperLink Magazine* is devoted to enriching the use of *HyperCard*. Each issue we present you with this column for the latest tips and tricks to make your stacks stand up and sing.

Window Shade

One of *HyperCard's* strong points is that it allows you to make graphics an integral part of your application, not just as static graphics, but interactive and animated presentations. Here's a routine to slide an opaque button up or down like a window shade so you can hide or show a graphic.

To illustrate we created a background with a picture of a woman waving (see Figure 1). On the card level we put a picture of a brick wall and window in it so the woman is visible. Then we added a background button called "Shade" that covers the edges of the window. The style of the button is set to "Rectangle" with "Show name" turned off. When it's in place, the woman in the window is completely covered.

Finally we added two buttons labeled "Open" and "Close." These buttons open (raise) the shade, or close (lower) the shade. We put these buttons just below the brick wall. Here's the script for the "Open" button:

```
on mouseUp
  get the rect of bkgnd button "Shade"
  put item 1 of it + 1 into finish
  put "," & item 2 of it + 1 after finish
  put item 1 of it + 1 into start
  put "," & item 4 of it - 1 after start
  choose button tool
  set dragspeed to 15
  drag from start to finish
  choose the browse tool
end mouseUp
```

Enter the following script for the "Close" button:

```
on mouseUp
  get the rect of bkgnd button "Shade"
  put item 1 of it  + 1 into finish
  put "," & item 2 of it + 155 after finish
  put item 1 of it + 1 into start
  put "," & item 4 of it - 1 after start
  choose button tool
  set dragspeed to 80
  drag from start to finish
  choose the browse tool
end mouseUp
```

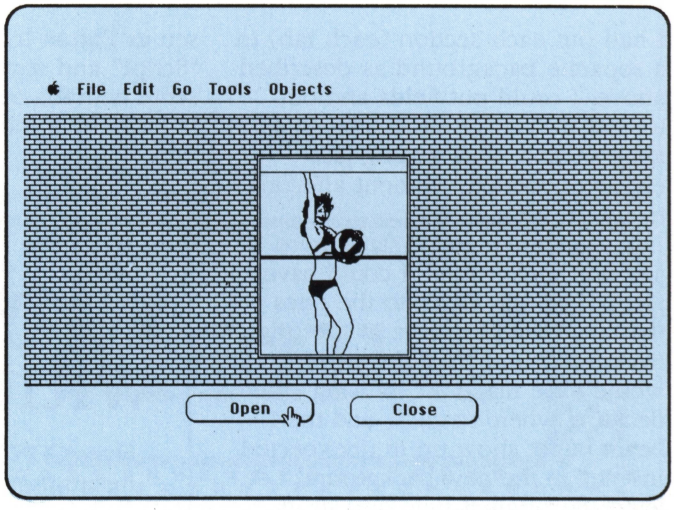


Figure 1

The window is shown in the open position. The button "Shade" covers the window when "Close" is pressed.

The two routines are similar. They both choose the button tool and use the `rect` function to get the coordinates of the button. Both routines also extract the value of the `start` variable from these coordinates in order to determine the point where the `drag` command should begin. The routines only differ in two ways. One is that we set the `dragspeed` to be faster when you close the shade. The other difference is what we place in the variable `finish`. It is set to the top (plus 1) for the "Open" button., In the "Close" button script it is set to a point 155 pixels below the initial bottom determined by the `rect` function earlier in the script.

Before you exit a routine your script should return the system to the same state in which it found it, we therefore `choose the browse tool` at the end. This

William K. Balthrop is Manager of Research and Development for HyperLink Magazine.

eliminates possible confusion for the user, and prevents the user from accidentally changing the stack.

Quick Script Selector

Have you ever hidden a field or button on your card, then realized you needed to edit the script? You have to remember its name or number, then use the message window to show it. When you're all done, you must hide the button or field again. This is time consuming, and if you can't find the hidden button or field, it could become the source of your next ulcer.

Here's a solution. By adding a button, a field, and a short script to your stack, you can have instant access to any card's field and button scripts. Place a button called "Edit Scripts" somewhere out of the way on your screen. Place a field called "TempField" in the center of the screen (See Figure 2) and hide it.

Now enter the following script for the card button "Edit Scripts":

```
on mouseUp
  repeat with x = 1 to the number of buttons
    put "B," & the short name of button x &¬
    return after List
  end repeat
  repeat with x = 1 to the number of fields
    put "F," & the short name of card¬
    field x & return after List
  end repeat
  put List into card field "TempField"
  show card field "TempField"
end mouseUp
```

This script is for the field "TempField":

```
on mouseUp
  GetLine
end mouseUp
```

Enter this next routine into the stack script so that it will be available throughout the entire stack.

```
on GetLine
  put the clickloc into C
  put the rect of card field "TempField"¬
  into R
  put textheight of card field "TempField"¬
  into H
  put trunc((item 2 of C - item 2 of R) / H)¬
  + 1 into L
  get line L of card field "TempField"
  if item 1 of it is "B" then
    edit script of button item 2 of it
  else
    if item 1 of it is "F" then
      edit script of card field item 2 of it
    end if
  end if
  hide card field "TempField"
end GetLine
```

When you click on the button "Edit Scripts," the field "TempField" appears with a list of all of the buttons and fields on your card. The way it is scripted here, it doesn't show background buttons and fields, but it wouldn't be hard to add. Now you can click on the name of the item within the field whose script you would like to edit, and the script editor appears letting you to change the script. After you're done editing the script, "TempField" is again hidden, leaving the screen as it was before clicking on the "Edit Scripts" button.

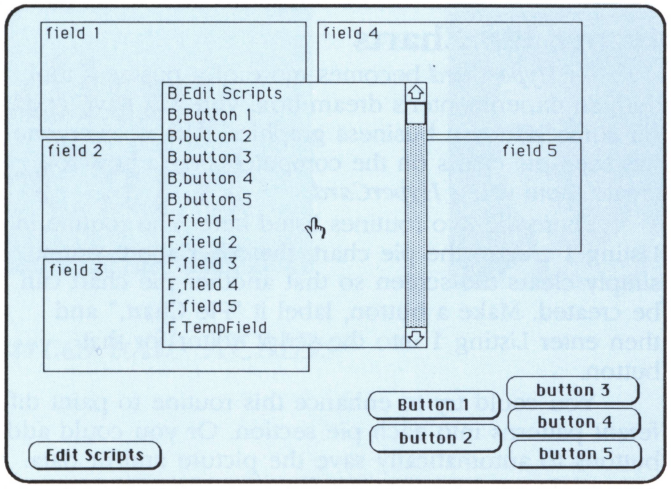


Figure 2

This screen shows "TempField" with its list of buttons and fields after the "Edit Scripts" button has been pressed.

Tips To Go

There are a number of occasions when using *HyperCard* that you will want to branch to a number of different stacks or cards depending on the user's response. A typical example would be when the user enters a choice or clicks on a selection from a menu. There are a number of ways this can be handled, but some tricky scripting may save you time and space.

The typical *HyperCard* approach to this problem would be to place a button on each choice in the menu. Each button would then have a `Go` command instructing it how to proceed. An alternative would be to place a transparent button on top of the list of options, then calculate which line of the field is being clicked on. You can use repetitive `If` statements as in the following script:

```
on OptionEntry
  global Option
  if Option is 1 then go to "Stack 1"
  if Option is 2 then go to "Stack 2"
  if Option is 3 then go to "Stack 3"
  if Option is 4 then go to "Stack 4"
end OptionEntry
```

There is a still more efficient method of branching when a list of options presents itself. This compact script does the same thing the previous script did:

```
on OptionEntry
  global Option
  go to item Option of "Stack 1,Stack 2,¬
  Stack 3,Stack 4"
end OptionEntry
```

Before calling this routine you must place the number of the option into `Option`, declare it a global variable in the calling routine, or simply include this procedure in the same routine which uses it.

Hyper Pie Charts

As *HyperCard* becomes more of a business tool than an experimenter's dream box, you will have need for some standard business graphics. Almost everyone has seen pie charts on the computer; here's how to create them using *HyperCard*.

There are two routines listed here. The routine in Listing 1 creates the pie chart; the other short routine simply clears the screen so that another pie chart can be created. Make a button, label it "Pie Chart," and then enter Listing 1 into the script editor for that button.

You could try to enhance this routine to paint different patterns into each pie section. Or you could add buttons to automatically save the picture and/or data. You might even want to get fancy and import data from another stack or a spreadsheet and display the number associated with each pie piece along with its percentage.

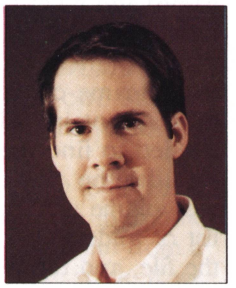
Here's the script for the "Clear Screen" button:

```
on mouseUp
  choose lasso tool
  doMenu "Select All"
  doMenu "Cut picture"
  choose browse tool
end mouseUp
```

Listing 1 - A script for a button that creates a pie chart

```
on mouseUp
  put "0,0" into finish
  put 0 into PieTotal
  put -Pi into Radians
  ask "Number of pie sections"
  put it into Pies
  repeat with x = 1 to Pies
    ask "Value of pie " & x & ":"
    put it into item x of WholePie
    add it to PieTotal
  end repeat
  choose oval tool
  drag from 125,50 to 375,300
  choose line tool
  put "250,175" into start
  repeat with x = 1 to Pies
    put (item x of WholePie / PieTotal)¬
    into Percentage
    add Percentage * Pi * 2 to Radians
    put trunc(sin(Radians)*125+250) into¬
    item 1 of finish
    put trunc(cos(Radians)*125+175) into¬
    item 2 of Finish
    put radians & return after field ¬
    "Test"
    drag from Start to Finish
  end repeat
  choose browse tool
end mouseUp
```

Xpanding HyperCard



Manipulating Resources with XCMDs

by James Paul

In the previous *Xpanding HyperCard* column, we explained that external commands (XCMDs) and external functions (XFCNs) are 68000 machine code generated outside of the *HyperCard* environment, and that they are called code "resources." We used an XCMD called *BarButton* to demonstrate how XCMDs allow you to add capabilities to *HyperCard*. We also provided detailed instructions on how to attach these code resources to stacks, the *HyperCard* application, or to your System file. Now that we've had some experience with an XCMD, let's look at an XFCN to see how it is similar to and subtly different from an XCMD.

The *ListRes* XFCN

We are going to explore an XFCN named *ListRes*, and see how it performs just like a regular *HyperCard* function. The *ListRes* XFCN can tell us the names of all the resources of a given type that are in a stack. It takes two parameters. The first is a complete Hierarchical File System (HFS) pathname for the stack we want to investigate; the second is the resource type we want to look for within that stack.

Here's the basic syntax:

```
ListRes(path,resType)
```

ListRes is an XFCN, not an XCMD. Just as XCMD stands for eXternal CoMmanD, XFCN stands for eXternal FunCtioN. Functions are used differently from commands. If a command, like `get`, is thought of as a verb that performs some action, a function can be thought of as a noun that can be put directly into a container. Some standard *HyperTalk* functions are `the date`, `the mouseLoc`, and `the target`.

You can receive information from the XCMD in the container called `the result`, but this is normally used to return an error code telling us the success or failure of an XCMD and is not as direct as using a function. In fact, `the result` is actually a function, as we will see in second.

Let's look at the differences between XCMDs and XFCNs when we use them in our script. If *ListRes* were an XCMD, which it's not, it would require two separate lines of *HyperTalk*:

```
ListRes path,resType
put the result into resList
```

The ListRes XFCN can tell us the names of all the resources of a given type that are in a stack.

ListRes, however, is an XFCN, not an XCMD, so we can do the same thing with only one line. This is how we really use *ListRes*:

```
put ListRes(path,resType) into resList
```

In other words, we can think of a function as not only a routine that we call, but also as a *HyperCard* container that gives us, what programmers call "returns," a value. One other important point is that *HyperCard's* own functions must either be preceded by the word `the`, or their parameters must be enclosed within parentheses as shown here. If the function is an XFCN or a user-defined function, it must have the parentheses even if there are no parameters being passed. The word `the` can only be used functions that are built-in to *HyperCard*.

Thus, you can see that not only is *ListRes* a function, but because `the result` returns a value in the same way, it is a function as well. In fact, if you write your own command in *HyperTalk*, you can place a value in `the result` using a return statement like this:

```
return errNum
```

Passing Parameters to a Function

Now that we see how to use a function in a script, we can look at the parameters for our XFCN, *ListRes*. We must tell it where to look for resources (a path-

James Paul is the chief programmer for Paul Software Engineering, and he is the author of the following XCMDs: ResCopy, DoList, KillResName, and the XFCN explained here, ListRes.

name) and what type of resource to look for (a resource type). A pathname is different from a file name. A pathname contains the file name, plus it contains all the information needed to find the file. A pathname includes the disk name (also called the volume name), followed by a list of folders (if any) that contain the file, and the file name at the end, with each of these elements separated form the others by a colon. For example, if we want resource information about a stack named "Our Stack" in a folder named "Stacks" in a folder named "HyperCard" on a disk named "Hard Disk," this would be our pathname:

```
Hard Disk:HyperCard:Stacks:
Our Stack
```

We can prove this to ourselves by opening "Our Stack," displaying the Message box and typing in `the long name of this stack` and pressing return. This is what we will see:

```
stack "Hard Disk:
HyperCard:Stacks:Our Stack"
```

Note that the Message box only has room for one line of text, so if you have long folder names, you may not be able to see the whole pathname in the Message box. By using `the long name` in our script, we can find out the pathname of our stack, wherever it is. We cannot, however, use `the long name` just as it is. We need to get rid of the word `stack` that comes at the beginning, and remove the quotes that surround the pathname. *HyperCard* puts these in so that `the long name` can be used as a *HyperCard* object. *ListRes* only needs to know the pathname itself. Here's the *HyperTalk* script to get just the pathname:

```
put the long name of this ¬
   stack into path
delete first word of path
delete first char of path
delete last char of path
```

The other parameter of *ListRes* is the type of resource we wish to look for in the stack. A resource type is always four characters long. Also, the names are case sensitive (i.e., upper case and lower case are *not* treated the same), so that there are more combinations

Listing 1 - Button Script that Plays all the sounds in a stack that uses the ListRes XCMD

```
on mouseUp
 set the cursor to 4
 put long name of this stack into path
 delete first word of path
 delete first char of path
 delete last char of path
 put ListRes(path,"snd ") into resList
 set the cursor to 4
 repeat with count=1 to the number of items in resList
 put "Now Playing:" && item count of resList into msg
 play item count of resList
 wait until the sound is "done"
 end repeat
 hide msg
end mouseUp
```

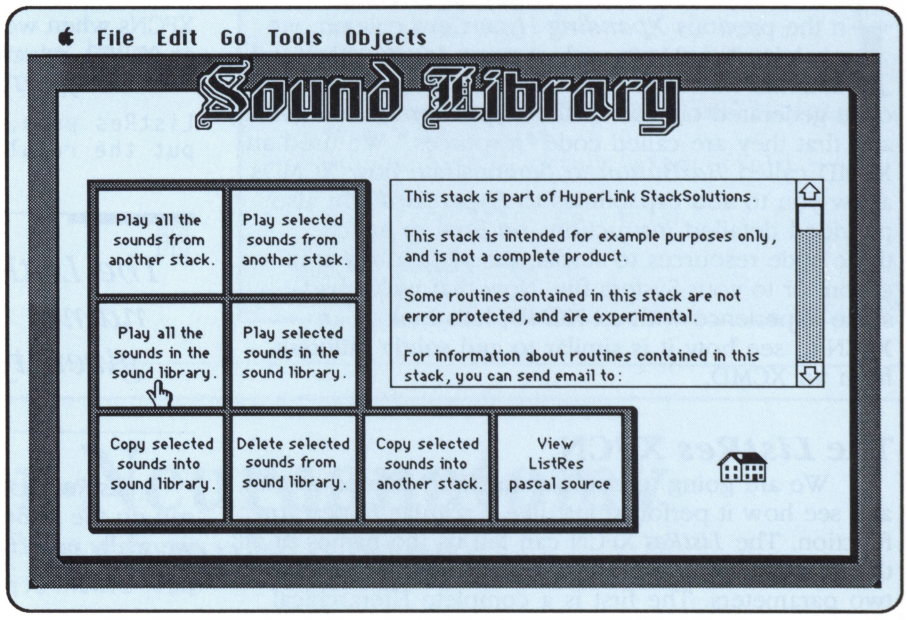


Figure 1

possible. We already know that external commands have the resource type "XCMD" and external functions are type "XFCN." Another common resource type in some stacks is the "snd " resource. Notice that there is a space at the end to make the resource type the required four characters long. These are sound resources which can be played with the `play` command from *HyperCard*. The "boing" sound is a "snd " resource, and is stored in the *HyperCard* application itself.

The Sound Library

Many stacks have their own sound resources which can be copied into other stacks. It would be nice to have a "Sound Library" stack with lots of sounds in it so that we can keep a collection of our favorite sounds. But how do we do it? Each time we add or remove a sound resource from our library stack we will have to make new buttons and change scripts, because each sound has a different name. How do we solve this problem? *ListRes*, of course. We can use *ListRes* to tell us the names of all the sound resources, so that we don't have to change our stack every time we add or remove a sound.

Let's use *ListRes* to write a button script to play, move, and generally organize all the sounds in a stack. Listing 1 shows the script that not only plays the sounds, but puts

Listing 2 - This button script prompts the user for the location of a file and then plays all the "snd " resources within that stack

```
on mouseUp
 put FileName("STAK") into path
 set lockscreen to true
 set lockscreen to false
 if path is not empty then
 put ListRes(path,"snd ") into resList
 if resList is not empty then
 push this card
 go stack path
 repeat with count=1 to the number of items in resList
 put "Now playing:" && item count of resList into msg
 play item count of resList
 wait until the sound is "done"
 end repeat
 pop card
 end if
 end if
 hide msg
end mouseUp
```

the name of each sound in the Message box while it's being played

Notice that the button we put this script into will be useful in our "Sound Library" stack because it plays all the sounds in the stack regardless of how many there are. We could keep our favorite sounds in this stack, and add or remove them at any time without having to change the script of our button.

What if we get a new stack with sounds in it, and we want to hear all of them? We can do this by changing our script a little bit, and by using the help of a public domain XFCN called *FileName*, which was written by Steve Maller of Apple Computer, instead of using `the long name` of our stack. The *FileName* XFCN lets us choose a file on our disk with a standard file dialog box. Even though it's called *FileName*, it returns a pathname, not a file name. Also, *FileName* lets us select the type of files to be displayed in the dialog box. Because we want to go to another stack, we will tell *FileName* to only show us stacks to choose from.

Because we want to play sounds that are in another stack, we have to go to that stack before we can play them. Here's what our script needs to do:

- Use *FileName* to choose a stack with sounds in it.
- Use *ListRes* to get a list of those sounds.
- Go to the selected stack.
- Play each of the sounds.
- Come back to our stack.

Listing 2 is the script that accomplishes these tasks.

Notice that we called our *ListRes* XFCN before we went to the other stack. Because *ListRes* is in our library stack, we can only use it when we are in our library stack, otherwise *HyperCard* will not understand it. Also notice that we are locking and unlocking the screen right after we use the *FileName* XFCN. This makes *HyperCard* update the screen to take care of the blank area left by the file selection box.

Now we know how to find the names of sounds that are in another stack, and we can play them from a script in our own library stack. If we want to move some of the sounds from the other stack into our library stack, we have to copy the "snd " resources we want from the other stack into our library stack.

A sample "Sound Library" stack with these scripts, as well as the Pascal source code for *ListRes,* is available in this issue's *StackSolutions* disk. The examples I've shown here are only part of the sound library stack, and are intended to help explain the basic concepts for its operation.

The sample sound library stack also uses some other external routines, one of which is explained in a stack named "HyperList," which is also included on the *StackSolutions* disk. "HyperList" explains how to use the *DoList* XCMD, which is used to select one or more items from a scrolling list.

Well, this concludes our tour of *ListRes.* If you have an idea for an XCMD or XFCN that you would like to see written, send me a letter.

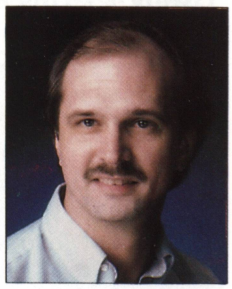
How to Contact Me

There are several ways to reach me:

- Write to me c/o *HyperLink* at P.O. Box. 7723, Eugene, OR 97401
- Send me mail on CompuServe or GEnie.

CompuServe: 72767,3436
GEnie: J.PAUL

Reviews of Reports and HyperCONTROL



Help for HyperCard Printing

reviews by Roger Wood

Product Name:	*Reports*
Company Info:	Activision, Inc. 3885 Bohannon Drive Menlo Park, CA 94025 (415) 329-0800
Price:	$99
Product Name:	*HyperCONTROL*
Company Info:	Nordic Software 3939 North 48th Lincoln, NE 68504 (402) 466-6502
Price:	$59

Ever since the release of *HyperCard*, one aspect of the program has been criticized more than any other—the lack of flexibility in printing options. The "Print Stack" standard allows for changing the size of printouts, but its all-or-nothing approach makes it impractical for any kind of selective printing of cards. "Print Card" lets you select cards, but not parts of cards, and is very time consuming when trying to print parts of stacks. Although "Print Reports..." offers some versatility (we even use it to print our mailing labels here at *HyperLink*), it cannot produce useful and attractive layouts of data from *HyperCard* stacks.

By making use of the *HyperTalk* command, `open printing`, it is possible to create some rather elaborate extractions from stacks, but as easy as *HyperTalk* is, it does not allow for the ease of use that has made the Macintosh famous in the field of desktop publishing.

In all fairness to Apple, the monumental task of creating enough printing options to satisfy the myriad users of *HyperCard* would have at best delayed *HyperCard's* release. Also, page layout software has demonstrated that print formatting is a complex operation that requires a specialized approach. Perhaps Apple has shown foresight by not trying to incorporate print formatting into an already very broad and deep application.

Third-party developers have rushed to fill this gap. Here, we look at two different support products that help bring new printing capabilities to stacks. *Reports* links stacks to an external report-layout application. *HyperCONTROL* employs an XCMD (external command) to expand printing capabilities of stacks.

Reports

Before we get into looking at this program's features, are there are a lot of them, a little history would be interesting. At the January MacWorld Expo in San Francisco, *HyperCard* was big news. A printing utility called *Reports!* from Nine-to-Five software was advertised before the show as being the answer to *HyperCard's* lack of printing versatility. By the end of the first day, the news was that Activision had signed a deal with Nine-to-Five to market the program and was demonstrating the new program. "It is not quite ready" was the word from Activision. Two months went by with the official word being "not yet."

The program was officially released on March 31, proving that *Reports* (Activision dropped the exclamation point) was not to fall into the vaporware zone. Instead a powerful, if somewhat complex, print manager has emerged. Does this mean that *HyperCard* is no longer plagued with gross printing limitations? With minor exceptions, this program does break down some major barriers.

The first release worked fine with *HyperCard 1.1*, but there were minor problems when the program was used with *HyperCard 1.0.1* (Apple's original release). By April 15, Activision announced an upgrade to *Reports* that would be sent automatically to all registered users. In addition, they put a special upgrade stack on CompuServe, GEnie, MacNet, and Activision's own Bulletin Board System.

By the time you read this, *HyperCard 1.2* will have been released and a new upgrade of *Reports* should soon be on its way to all registered owners. Mike Long of Nine-to-Five, *Reports's* programmer, has informed us that as long as owners send in their registration card, they will receive the promised automatic upgrade including the latest version of *HyperCard* on the *Reports* disk. All in all, these different upgrades may seem inconvenient to many *Reports* users, but we feel that such support should be applauded. Indeed, the fact

Roger Wood is the Editor of HyperLink Magazine.

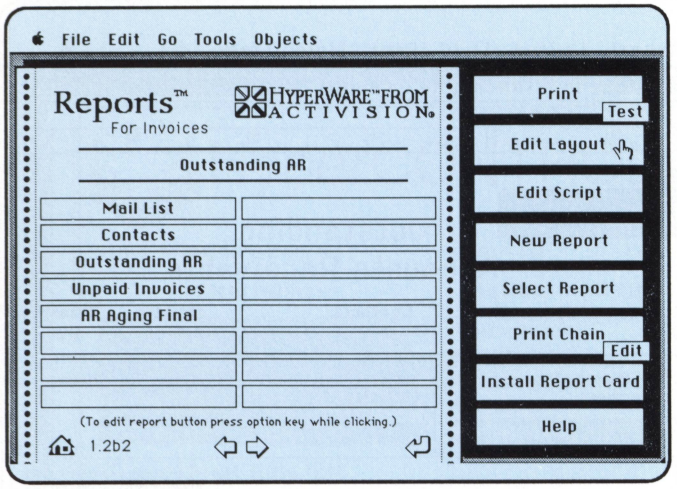


Figure 1

The "Report" card must be installed in any stack that you wish to use with Reports, in this case a stack called "Invoices." The names of different reports are displayed in the rectangular fields. The report layout displayed between the horizontal lines at the top of the screen, in this case "Outstanding AR," is the one that will be printed, edited, etc. when you press one of the buttons along the right side.

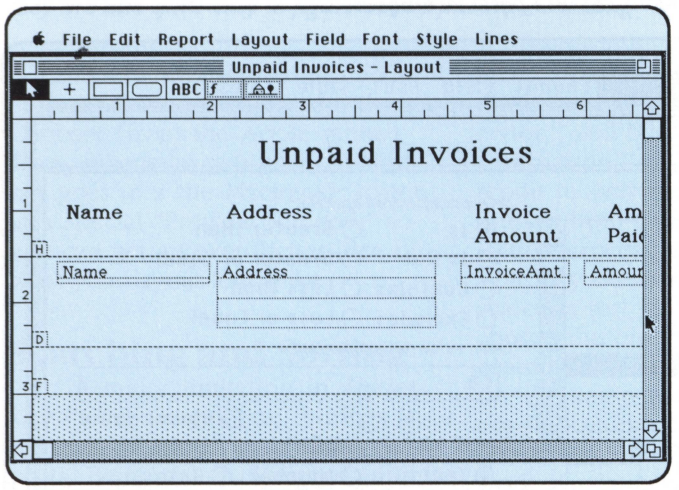


Figure 2

The Reports editor screen lets you combine background field info and labels into coherent and well thought out reporting formats.

that Activision will be sending out the latest version of *HyperCard* is a service to their users by, ensuring product compatibility.

Worth Waiting For

Reports is powerful—but with power comes complexity. Although many of the program's tools can be used intuitively, it takes some learning. As we stated earlier, page layout and report generation are not trivial jobs. This leads me to one other comment: this review can in no way be exhaustive. The manual is quite long and complete, but even it leaves some areas begging for more examples. I will try to show you the most important the features of the package without delving into all the scripting options and complex searching and sorting functions. These are the very features, of course, that give *Reports*, and thus *HyperCard*, a lot of depth.

Reports is a stand-alone application that runs outside of *HyperCard*. It links to and from *HyperCard* via a "Report" card (see Figure 1) that is installed into every stack with which you wish you use *Reports*. This card uses several XCMDs and XFCNs that must be installed into the "Home" stack or the stack that is going to use *Reports*. All installation, whether of commands or the "Report" card, is handled quickly and easily by a special "Installer" card.

One important pitfall when installing: If you go through the installation process and immediately try to use *Reports*, it doesn't work. Even if you read the 210 page manual, you will meet with great frustration unless you also read the "*Reports* Update" sheet that accompanies the program. This tells us that "after installing the *Reports* commands in your Home stack for the first time, you must quit and restart *HyperCard* to initialize the freshly installed resources." This is the way *HyperCard* works with its XCMDs, and not, Activision assures us, a function of *Reports*. It's too bad *Reports* didn't have a message right in the program to quit and restart as part of the installation process—it would ensure a smoother installation process.

Once the commands are safely installed, you might be tempted to try using the program on a stack of your own. We don't recommend this unless you are a very quick learner. Instead, work through the nicely paced tutorial that goes through the program's many features. Don't plan to buy *Reports* today and get that database report whipped into shape by "yesterday."

Creating and Editing a Report

Once you've completed the installation procedures, it is time to edit a layout. You can either choose "New Report" to start a new layout, or select "Edit Layout" to work on the existing layout, whose name appears between the horizontal lines near the top of the card. In the former case, you are whisked away to the layout editor, which is the *Reports* application itself, and you are asked to choose the stack and the background of the stack that you wish to work with. This points up an important aspect of the program that may not be apparent to newcomers to *HyperCard*: The background, not the stack, is the lowest common denominator for a report layout. This means that you can only use one background for a each report. If a stack has several backgrounds, you will have to have separate reports for each one. Because background fields are the *HyperCard* containers that hold related information within a stack, this makes sense.

Layout Sections

For maximum flexibility in creating and working with layouts, *Reports* divides each report layout into sections. Figure 2 shows the three most commonly used sections in a layout, Header, Detail, and Footer. These sections are identified by the first letter of the section type in the lower-left corner of each section. All of these sections are optional, although the Detail section is practically essential to any meaningful report, because this is where the information from your stack's fields will be placed. The Header and Footer sections

are equally important to allow for labeling, page numbering, and so on for every page of your report. There is also a Total section available (included in the layout shown in Figure 4). This section is specialized for working with the Detail section, and allows automatic totaling and subtotaling of fields containing numeric values.

There is one other section type called Break not shown in either example. The Break section allows you to force page breaks based upon logical groupings of data such as date, amount, etc.

The Tools

A major Macintosh innovation is the ability to print using various fonts and graphics. This was definitely lacking in *HyperCard's* built-in "Print Report..." option, but it is *Reports's* strong suit. These elements are created and manipulated using the tools across the top of the screen directly beneath the title bar (see Figure 3). The four on the left are relatively familiar to most users. They are called (from left to right) Pointer, Straightline, Rectangle, and Rounded Rectangle. The Pointer is for moving objects around, and the other three are for creating graphics.

The fifth tool (ABC) is called the Text Field tool and is for creating static text as labels for your layout—e.g., the words "Outstanding Accounts Receivable" and the column headings in Figure 3. The tool with the little "f" in it is called the Information Field tool. Generally, these fields correspond to background fields within the stack—they have to have the same name. In the case of Figure 3, "Invoice #," "Name," "Phone," and "Contacts" are all fields within the "Invoices" stack. You may also use these information fields for global values, such as the date, that do not correspond to a field within the stack.

The tool on the far right is called the Graphic Field tool, which allows you to import graphics using scripts to copy and paste them directly from the artwork within stacks. This gives us an idea of how scripts can be used in *Reports*. Suffice it to say, this is just one of many features in this program that is not for beginners. More advanced *HyperTalk* enthusiasts, however, will find the extensibility of *Reports* a great plus.

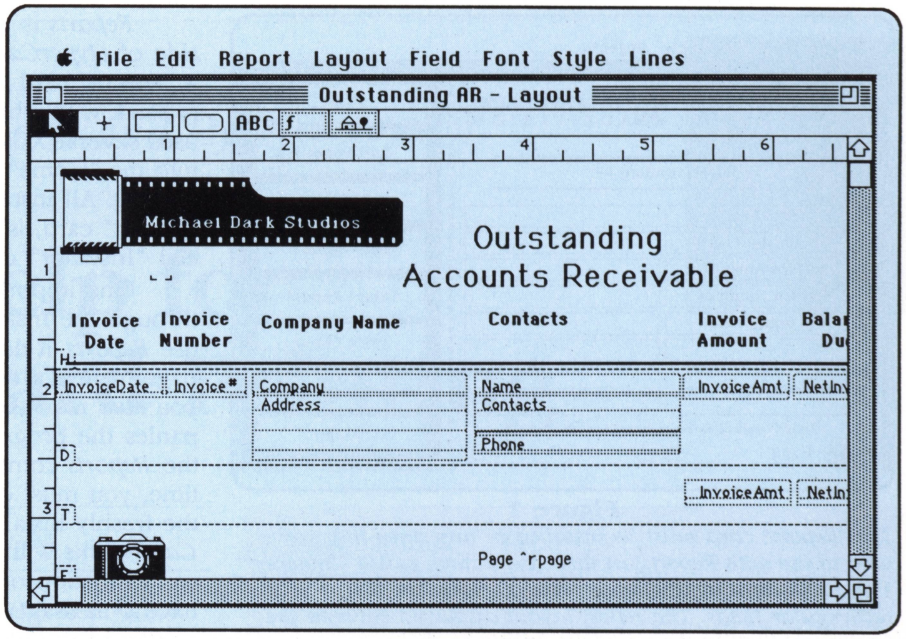


Figure 3

This layout is more complex than the one in Figure 2. It includes a Totals field that automatically figures the subtotals and totals for a report. Also graphics were added using standard Macintosh cut and paste techniques.

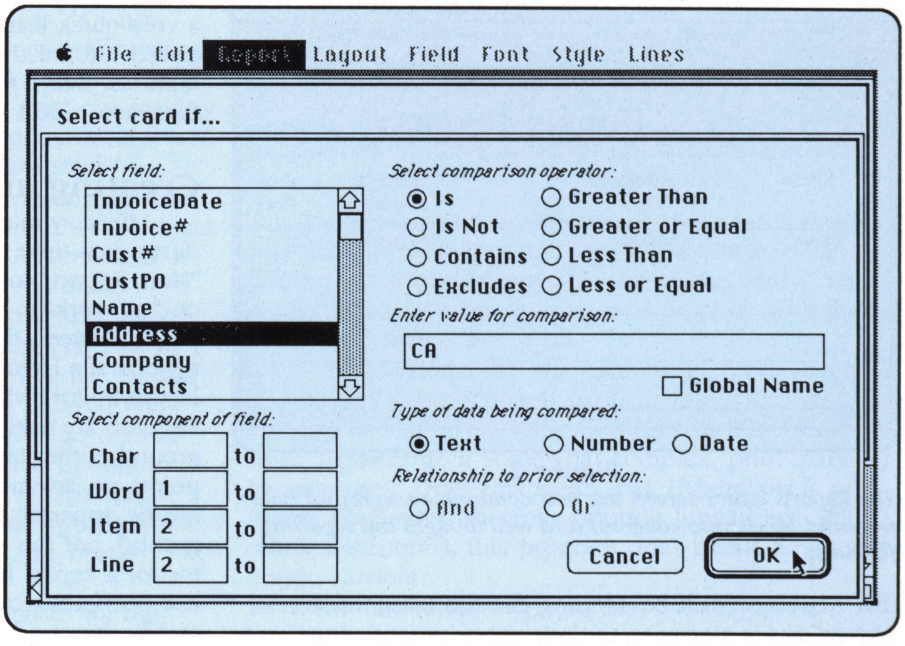


Figure 4

This "Select Card if..." dialog box makes selective printing very easy. The "And" and "Or" functions come into play when more than one item is put into a list of more than one search parameter.

The tools conform well to Macintosh user guidelines, and we found that if you can use an object-oriented program like MacDraw, this editor presents no problems.

Printing options might seem confusing at first, but prove to be very flexible. When in the *Reports* editor, you only print the layout. To actually print the report you quit the editor and return to *HyperCard*. To print the report you select it and press "Print" on the "Report" card (see Figure 1).

One handy addition is a product called *Preview* from Software by Design that is included with Re-

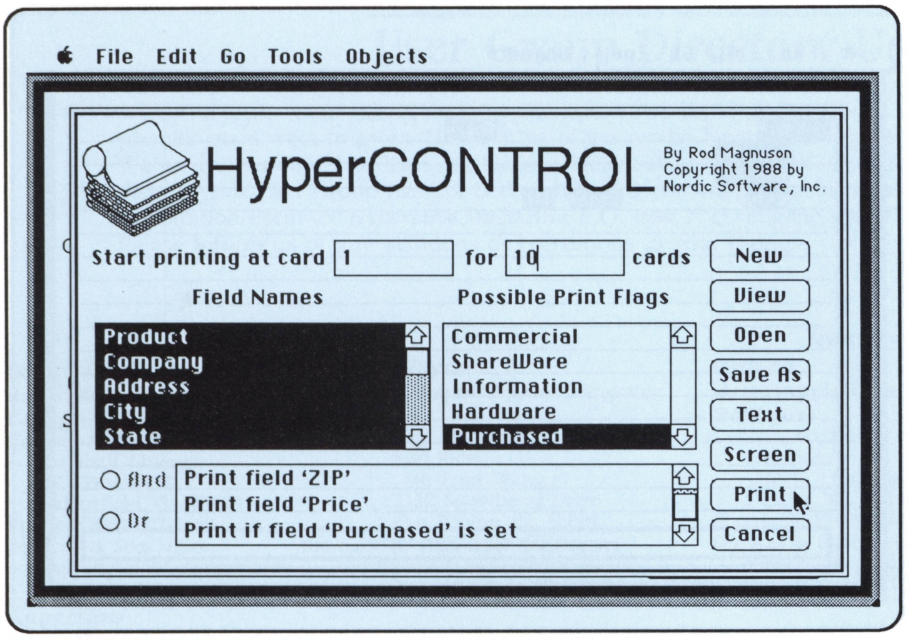


Figure 5

This dialog box appears when you click on the HyperCONTROL button installed in the background of a stack. The manual makes all the functions quite clear, but they are also quite easy to figure out by just experimenting.

ports. You copy the file into your System folder and select it with Chooser (from the Apple menu). Then, whenever you print, the output goes to a the Macintosh screen in a special "Preview Window" so you can get an even better idea of what your layout looks like.

Searching and Sorting

A major limitation in *HyperCard's* built-in printing routines is the lack of selectivity in doing reports. Being able to keep an entire database in one stack and being able to print out only certain records is essential. *Reports* supplies this capability admirably. Not only can you print selectively based upon the value of a field, but you can limit the selection by using one of eight "comparison operators" (see Figure 4) and can choose the data type to be either "Text," "Number," or "Date." You can be even more selective by limiting the component of the field using *HyperTalk's* familiar "chunk" expressions to pick out a particular character, word, item, and/or line of a field, or a range of these. For example, in the "Select Card if..." dialog box shown in Figure 3, by selecting the word 1, item 2, and line 2 of the "Address" field, the choice can be based upon the state of a particular record. Then by picking the "Is" radio button as the "comparison operator," and "CA" as the "value for comparison," we have limited this report to only print cards with addresses from California. But, the selection process needn't stop here. You may do more than one such selection test for a report and choose the order the tests are executed. You may also use the "And" or "Or" radio buttons (under "Relationship to prior selection")in order to control how a given test will work in relation to the others. The number and order of the different selection tests can be easily edited using a related dialog box that lists the tests with descriptive names.

Using the sort functions available within *Reports* is quite similar to the selection process in that you can not only select a field, but also one particular component. This is achieved by using the same chunk expression devices employed in selecting. Also, a list of sort functions is maintained so you can do multi-level sorts. For instance, you can first sort by state and then sort by city, and the sort will know to leave the cities from one state together. For better or worse all sorting done within *Reports* has no effect upon the stack. It would have been nice if the sorting could have optionally affected the stack order.

It should be noted that these searching and sorting functions go well beyond those available within *HyperCard*, especially in terms of supporting combinations of "And" and "Or" choices within the Select function. Also the ability to do multi-level sorting is difficult to duplicate in *HyperCard*.

Scripting and Beyond

Although the details of *Reports's* scripting options are too many to list in this review, they *are* extensive. All *HyperTalk* scripting is available, plus a number of new functions are built-in. Scripts can be called before or after a report is printed as well as before a particular section prints. Our tests showed that the scripting interface was handled smoothly and appeared to work as documented.

There is really only one thing that seemed to be lacking in the way *Reports* lets you build your report. There is no simple way to select card buttons as the criteria for selecting cards. For example, you might have a check box for a particular type of customer, say, wholesale buyer. In *Reports*, as with *HyperCard*, you must write a script that can check the `hilite` property of the button and decide whether to include or exclude a card from a particular operation. This one inconvenience is very minor, however, and is more than outweighed by the great features included in this excellent product.

HyperCONTROL

While *Reports* was getting all the attention, Nordic Software released a print utility in the form of an XCMD called *HyperCONTROL*. Working totally within *HyperCard*, here's what it doesn't do:

- No graphics printing or fancy formatting.
- No font changes. (It prints in 9 point draft quality on the ImageWriter and 9 point Courier on the LaserWriter.)
- It adds no sorting routines.

To Nordic Software's credit, all of these shortcomings are explained up front in the short (20 pages) but very clear manual.

What *HyperCONTROL* can do, however, goes a long way toward answering many limitations of *HyperCard's* own printing features.

The really good news is that *HyperControl* is very easy to use. You install it by using an "Install HyperCONTROL" button to put the XCMD in your "Home" stack. It even tells you right in the message box to quit and restart *HyperCard* in order to make use of the newly installed XCMD. It's in the manual, too, but this kind of clarity is great. Then all you do is copy the "HyperCONTROL" button into the background of the stack you want to use it with. When you click on this button, the dialog box in Figure 5 appears on the screen.

As with *Reports* background fields (which must have names) can be selected or left out of any report. *HyperCONTROL's* string search function lets you be more selective based on whether a particular text string is in one of your fields. The string to be searched for is easily selected by just double-clicking on the field and typing the string into a dialog box. If you choose two or more search strings to select whether a card is included in the report, you have an additional degree of selectivity. You may choose to include only the cards that contain all the search strings (using the "And" radio button), or to include those cards that contain any of the chosen strings (using the "Or" button).

The "Possible Print Flags" list on the right side of Figure 5 allows you to select cards based upon the condition of card check-box buttons or radio buttons. For example, you can choose to include a card from a report if a particular card button's `hilite` property is true or not (i.e., if it is checked or not). The package even provides a "HyperDisperse" button to make it easy to put a new card button on every card in a stack—a real time saver. (Don't worry, there's a "HyperUndo" to get rid of buttons, too.)

The Layout

When you have selected the different criteria for your report, you then click "New" and the layout form (see Figure 6) appears providing you with the means to do your layout. Although this is not as versatile as *Reports's* editor, its graphic nature makes it much easier than *HyperCard's* built-in "Print Report..." option to place the fields where you want them. You can move different fields anywhere on the screen, and even put several fields on the same line. Again, although not that flexible, it is easy and gives you quite a bit of choice as to how a layout will look.

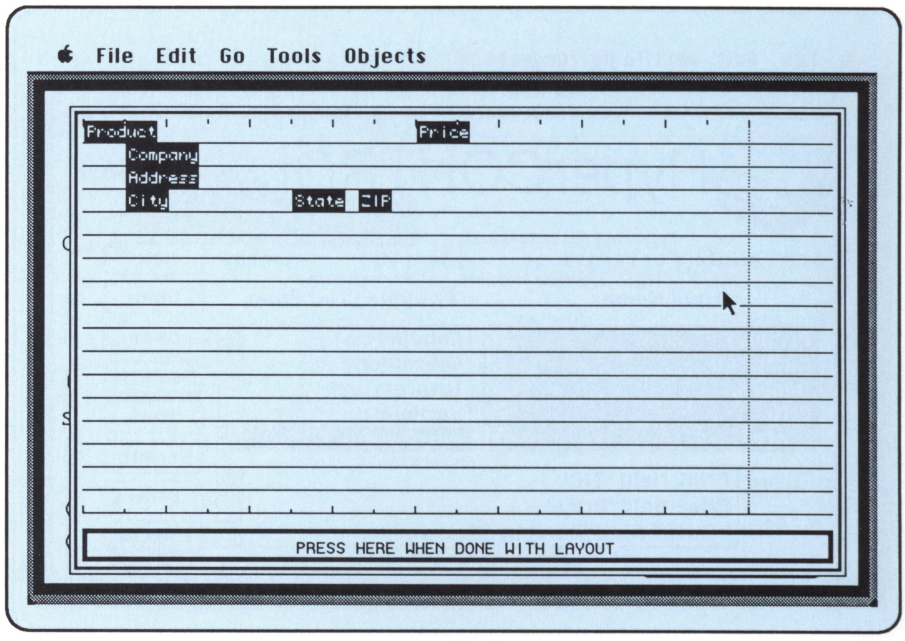


Figure 6

When you create a report using the "New" button, this layout sheet appears, and any of the fields in the report may be moved to create a custom layout. This allows for easy rearranging of fields.

Once you've completed the layout and clicked on the button at the bottom of the screen, you can use the "Save As" button to save this particular configuration on disk. The only part of the this configuration that is not saved is the range of cards to be printed. This must be set each time you reload—a bit of an inconvenience.

In addition to the layout window, there is also a "Preview" button in the *HyperCONTROL* dialog box (see Figure 5). When you click this, you get to see the actual records that you are formatting—a great debugging aid when putting your report together.

Print to a Text File

One of the truly great features of *HyperCONTROL* is the ability to send the printer output to a text file on disk. You just press the "Text" button, and a "Standard File" dialog box appears for you to name the file. This means that it serves a function many have demanded—exporting data in a selective fashion. With a little thought this feature could be used to export data to many database programs, and certainly to word processors and page layout programs.

Formatting of these text files is quite intuitive. Anytime you put a field on a new line, a return character is placed in the file. If more than one field is put on a line, a tab character is placed between them.

And Totals, Too

Finally, *HyperCONTROL* rounds out its package with a simple but very useable totals function. The formatting here is simple, either with dollar signs or without, with decimal points aligned in the former case.

All in all, *HyperCONTROL* is a fine application. It's not the full-blown report generator that *Reports* is, but it provides essential functions not available in *HyperCard*. In addition, such an extensively featured utility in the form of an XCMD is great to see. It provides a great example for many serious programmers who want to place products in the *HyperCard* market.

User Group Directory Update

Welcome to the User Group Directory. Last issue we listed all the groups who had registered by the time the premier issue went to press. Here's a list of those who have either registered since then or sent us changes. We've made a complete list available as a stack called "HLM MUG Directory" on this issue's *StackSolutions* disk. If your User Group has not been listed, and you would like to register with *HyperLink*, send for a Registration Form to: *HyperLink Magazine*, P.O. Box 7723, Eugene, OR 97401, Attention: User Group Editor. Please inform us of any additions or corrections to your listing.

CALIFORNIA

Bakersfield
Bakersfield Apple Users Group
Contact Address:
Marshall Moens
P.O. Box 2802
Bakersfield, CA 93303
Phone: (805) 872-7463
Services & Specialties:
"Apple Patch" newsletter

Cupertino
Apple HyperCard User Group
Contact Address:
David Leffler
20525 Mariani Ave.M/S27AQ
Cupertino, CA 95014
Services & Specialties:
"Windoid" newsletter, lectures, demos

Roseville
International Blind Users Group
Contact Address:
Otto Haiungs
P.O.Box 1352
Roseville, CA 95661-1352
Phone: (916) 783-0364
Services & Specialties:
BBS for the blind, CompuHelp™, (916) 786-3923

Sacramento
CalTrans MUG
Contact Address:
Stephan C. Prey
1120 'N' Street Room 4325
Sacramento, CA 95814
Phone: (916) 445-3229
Services & Specialties:
Newsletter, lectures. Group for CalTrans employees only.

San Diego
San Diego MENSA Apple U.G.
Contact Address:
Tom Wade
6030 Fennell Ave.
San Diego, CA 92114-4112
Phone: (619) 263-1213
Services & Specialties:
Newsletter, lectures, demos

ILLINOIS

Chicago
American Bar Association MUG
Contact Address:
Lawrence Husick
c/o Daniel Kegan
Chicago, IL 60603-4969
Phone: (215) 265-6666
Services & Specialties:
Newsletter (quarterly) meetings (quartlerly), BBS via ABAnet, software

INDIANA

Ft. Wayne
Fort Wayne Apple Computer U.G.
Contact Address:
Ran Rice
c/o Fred Warner
7112 Snowberry Drive
Fort Wayne, IN 46804
Phone: (219) 749-9705
Services & Specialties:
Newsletter, meetings, demos

MASSACHUSETTS

Boston
Boston Computer Society
Contact Address:
Martha Healy
One Center Plaza
Boston, MA 02108
Phone: (617) 367-8080
Services & Specialties:
Newsletter, lectures, demos, BBS

MISSOURI

Jefferson City
apple JAC M.U.G.
Contact Address:
Thomas R. Piper
2539 Lexington Drive
Jefferson City, MO 65101
Phone: (314) 634-3102
Services & Specialties:
Newsletter, lectures, public domain library, HyperCard library, demos, BBS (314) 636-6502 nights and weekends

NORTH CAROLINA

Wilmington
Macintosh Apple Corps of Wilmington
Contact Address:
Roger Pozig
2840 South College Road
Suite 320
Wilmington, NC 28403
Phone: (919) 392-5262
Services & Specialties:
Newsletters, presentations, public domain software library, ribbon inking, new users SIG

NEBRASKA

Omaha
Metro Apple Computer Hobbyists
Contact Address:
Joe Davis
3506 N. 113th Plaza, Apt.# 1
Omaha, NE 68164
Phone: (402) 339-7619
Services & Specialties:
Newsletter, product and programming demos

New Brunswick
Amateur Computer Group of New Jersey
Contact Address:
Keith Sproul
698 Magnolia Road
New Brunswick, NJ 08902
Phone: (201) 821-4828
Services & Specialties:
Demos, lectures, very good PD library, including HyperCard

NEW JERSEY

Princeton
Princeton Macintosh Users' Group
Contact Address:
Richard H. Williams
C-427 Engineering Quatrangle
Princeton University
Princeton, NJ 08544
Phone: (609) 397-8438
Services & Specialties:
BBS (609) 921-1523

NEW YORK

Binghamton
Southern Tier Apple Core
Contact Address:
Bob Marean
c/o Binghamton City School District
98 Oak Street
Binghamton, NY 13905
Phone: (607) 773-4745
Services & Specialties:
Newsletter, lectures, demos, pd library

Seneca Falls
The Apple Corps
Contact Address:
Tim Davies
55 Stevenson Street, Apt. 5
Seneca Falls, NY 13148
Phone: (325) 568-9718
Services & Specialties:
Newsletter, demos, Mac pd library, classes, hot-line help

Whitesboro
Upstate Apple Users Group
Contact Address:
George Engel
c/o The Computer Factory
99 Commercial Drive Route 1, Box 17-A
Whitesboro, NY 13492
Phone: (315) 336-8060
Services & Specialties:
Newsletter, lectures, demos

OHIO

Canton
MAC2
Contact Address:
Robert Funk
P.O. Box 8081
Canton, OH 44711
Phone: (216) 455-6387
Services & Specialties:
Newsletter, public domain libraries, lectures, demos, group purchase discounts

Wooster
Country Computer Club
Contact Address:
Scott Barta
2175 E. Messner Rd.
Wooster, OH 44691
Phone: (216) 264-2494
Services & Specialties:
Newsletter, BBS, lectures, demos, group purchases

VIRGINIA

Fredericksburg
Rappahanock Apple Group
Contact Address:
Joe McAllister
6014 Battlefield Green Drive
Fredericksburg, VA 22401
Phone: (703) 786-6577
Services & Specialties:
Newsletter (monthly), demos, field trips

WASHINGTON

Pullman
Palouse Area Microcomputer Assn.
Contact Address:
Ed Steever
Box 2149 C.S.
Pullman, WA 99165-0903
Phone: (509) 335-9531
Services & Specialties:
Lectures, demos, video tape, Mac user group connection info from Apple

Letters to the Editor

—continued from page 9

in the future. This magazine also makes *HyperCard* a pleasure to use. I was a bit in the dark as to how to get much from it before. Please keep up the good work.

I would like to see someone write a simple accounting program using *HyperCard* that would print out a few basic reports. I guess you wonder why I don't do this myself, but I am not that well acquainted with the program as yet. Maybe someone who has been experimenting with it could come up with something generic that would be of use to the rest of us out here, with a minimum of fine-tuning.

I am impatiently looking forward to the next issue. By the way, are you going to institute a program whereby a person could subscribe for all the disks in advance, without having to order each one after the magazine comes out?

Orva J. Cowan
Vancouver, WA 98665

Thanks for all the positive feedback, Orva. As you can see from our new order form on the inside back cover, we now offer disk subscriptions. A year's subscription to both magazine and disk is $65. If you are already a subscriber to the magazine you can upgrade to a disk subscription for the difference—just $40.

Get Technical

I have received your first issue and thoroughly enjoyed reading it from cover to cover. I am looking forward to learning more advanced techniques in your future issues. I believe you should aim your magazine at more advanced users as all the other Macintosh magazines have HyperCard sections that will appeal to the beginning user.

Steven Robertson
Apollo Software
Kent, WA

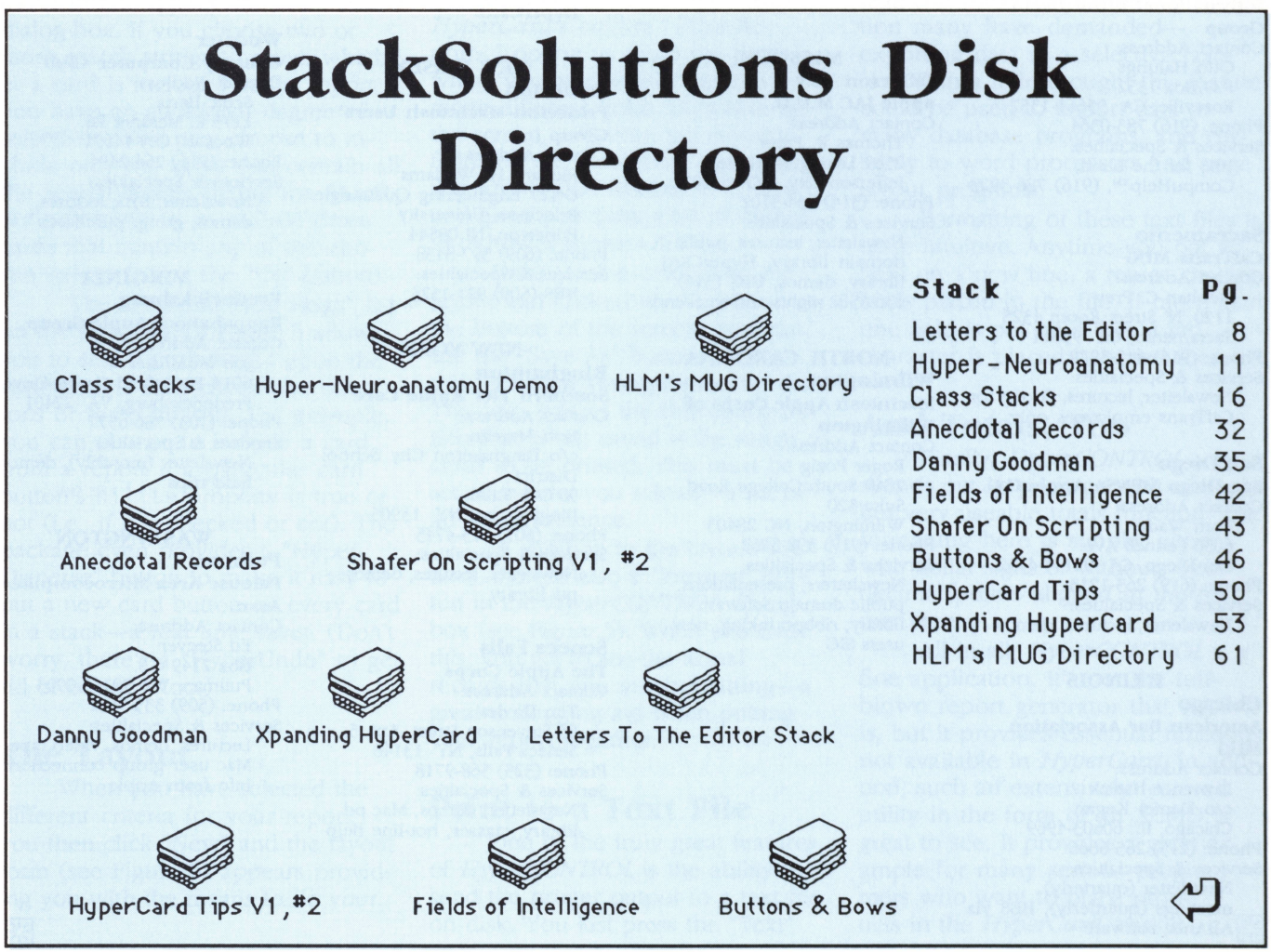
Glad you enjoyed the premier, Steven. I hope our columns and features fill the bill for you—and for the next reader, too.

Filling The HyperVoid

I have had my Mac SE for only two weeks, and I am a novice at *HyperCard*. I am using Danny Goodman's book. Your publication really fills a great void. Thanks, and keep 'em coming!

Frank Vermeersch
Dallas, TX 75381

You're welcome, Frank, and we will keep 'em coming every other month.



HYPERLINK MAGAZINE

The premier HyperCard information magazine!

As a new subscriber to **HyperLink Magazine**, you may take advantage of our special offer to receive the **StackSolutions** microdisk for your first subscription issue — **FREE!** This **StackSolutions** microdisk has all the custom stacks, buttons, intelligent fields, scripts, and graphics described in your first issue of **HyperLink Magazine**.* **Please note:** The $2.95 Shipping & Handling **must** be paid **with** your magazine subscription to take advantage of this offer. Subscribe TODAY! Mail the subscription form and payment to:

HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401.

In a hurry? Place your order by calling our toll-free subscription line at 1-800 544-0339. Please have your VISA or MasterCard handy.

Subscribe Now!

SUBSCRIBER NAME _____ PHONE (____) _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

METHOD OF PAYMENT: ☐ VISA ☐ MasterCard ☐ Check (enclosed)

Credit Card Number _____ Credit Card Expiration Code _____

Signature of Card Holder _____

CHOOSE one of the following HyperLink Magazine subscriptions —

☐ One Year $25 ☐ Two Years $45 ☐ Three Years $60 (HyperLink is a bi-monthly publication)

☐ Send the free premium StackSolutions disk. Enclosed is the $2.95 Shipping & Handling charge.

OR CHOOSE a combination HyperLink Magazine/StackSolutions Disk subscription —

☐ One Year $67.95 ☐ Two Years $124.95 ☐ Three Years $172.95

(The combination subscription prices include $2.95 S&H for the first disk, which is free.)

ALSO INCLUDE —

☐ The Vol.1 #1 HyperLink Magazine back issue (if available) for $4.95

☐ The Vol.1 #1 StackSolutions disk (includes S&H) for $14.95

☐ The Vol.1 #2 (this issue's) StackSolutions disk (includes S&H) for $14.95

Total amount enclosed for the checked items $ _____

Type of machine: ☐ Mac+ ☐ Mac SE ☐ Mac II ☐ Other _____

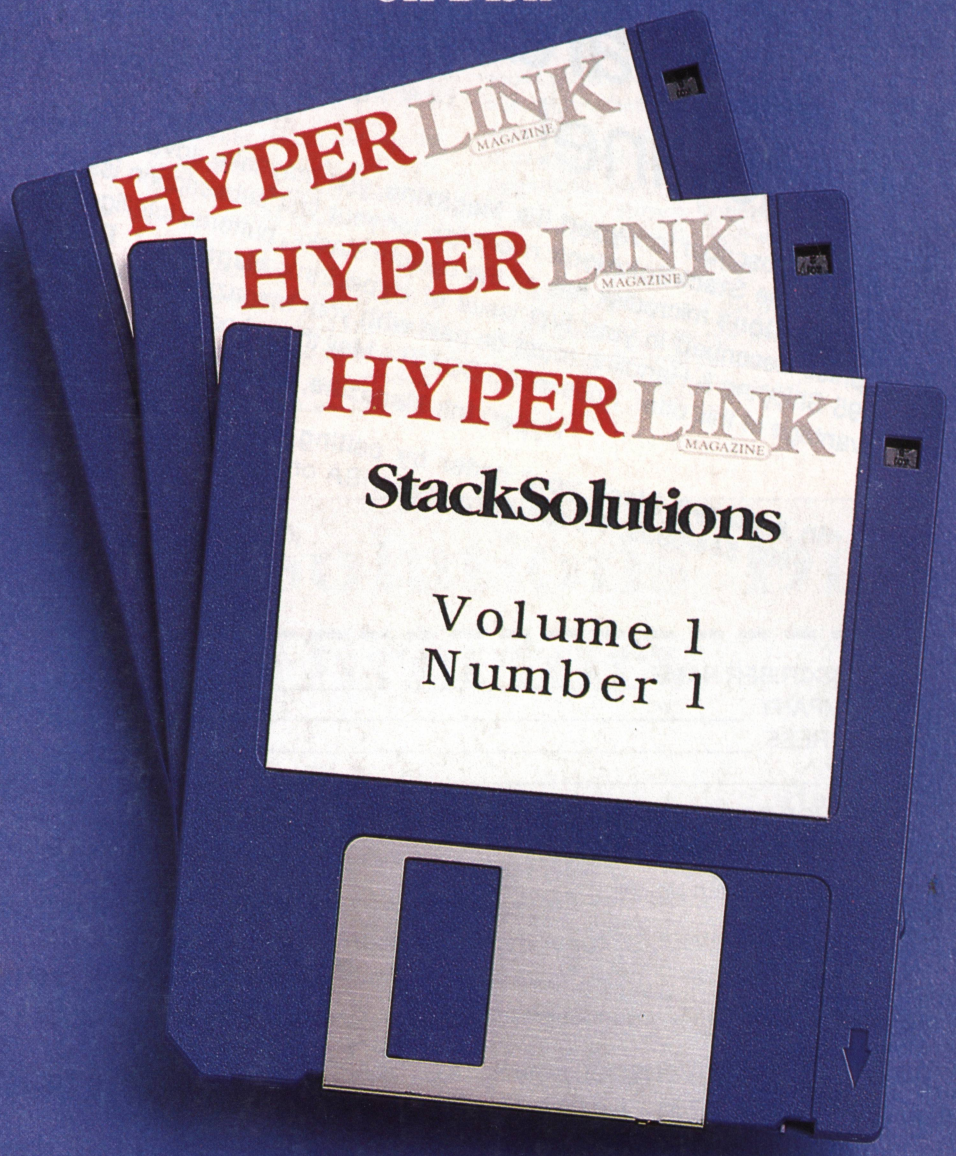
Memory size: _____ Do you have a hard disk? _____ If so, what size _____

Comments? _____

* Please note that the premium (StackSolutions microdisk) has a retail value of $12. If for any reason a refund is to be made on this subscription, the value of the premium and any delivered issues are subtracted from the refund.

HyperLink Magazine Presents StackSolutions

on Disk



We try to put every bit that we can into the pages of **HyperLink**, but we still find portions of stacks, such as sound and some graphics, impossible to print. In addition, there are cases when we don't have room to print large stacks even though we describe them within the magazine. Our solution is to put everything covered within an issue of **HyperLink** on a **StackSolutions** disk. Every issue of **HyperLink** has a companion **StackSolutions** disk. You may order an individual **StackSolutions** disk or a combination **HyperLink Magazine/StackSolutions** subscription using the order form inside the back cover.